Real attacks on virtual networks: Vivaldi out of tune

Mohamed Ali Kaafar
Planète Project
INRIA Sophia Antipolis
mohamed.kaafar@sophia.inria.fr

Laurent Mathy
Computing Department
Lancaster University
laurent@comp.lancs.ac.uk

Thierry Turletti
Planète Project
INRIA Sophia Antipolis
turletti@sophia.inria.fr

Walid Dabbous
Planète Project
INRIA Sophia Antipolis
dabbous@sophia.inria.fr

***Abstract*— The recently proposed coordinates-based systems for network positioning have been shown to be accurate, with very low distance prediction error. However, these systems often rely on nodes coordination and assume that information reported by probed nodes is correct. In this paper, we identify different attacks against coordinates embedding systems and study the impact of such attacks on the recently proposed Vivaldi decentralized positioning system. We present a simulation study of attacks carried out by malicious nodes that provide biased coordinates information and delay measurement probes. We experiment with attack strategies that aim to (i) introduce disorder in the system, (ii) fool honest nodes to move far away from their correct positions and (iii) isolate a particular node in the system through collusion. Our findings confirm the susceptibility of the Vivaldi System to such attacks.**

I. Introduction

Recent years have seen the proliferation of application-level overlays (or overlays in short) to support many different types of applications ranging from file sharing to VoIP (e.g. [1] [2] [3] [4], etc). 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 neighbour selection during overlay construction and maintenance. Despite efforts to keep proximity measurements to a minimum on many overlays, the simultaneous presence of several overlays can result in significant bandwidth consumption by proximity measurements (i.e. ping storms) carried out by individual overlay nodes [5]. This problem is also compounded by dynamics in overlay membership, as measuring and tracking proximity within a rapidly changing group can prove very onerous.

To avoid such overhead, the idea of distance estimation and network positioning/coordinate systems were introduced. In such systems, the thesis is that if each node can be associated with a "virtual" coordinate in an appropriate space, distance between nodes can be trivially computed without direct measurement. In other words, as long as a reasonably accurate position for a node can be obtained with little effort, much of the distance measurement sampling cost can be eliminated and the remaining overhead amortized over many distance predictions.

Most of the recently proposed coordinates-based systems have been shown to be accurate, achieving very low prediction error. On the other hand, a robust, stable, scalable and low overhead coordinate system can often only be realized at the expense of slow convergence times. In such a scheme, new nodes joining the system only reach a good estimate of their own coordinates after a lapse of time in the timescale of tens of seconds to several minutes. Such convergence times, which are longer than those typically achieved with individual sampling of distances by nodes, are often unacceptable for applications and this argues for a deployment of coordinate systems as a service: every host could run a coordinate system daemon at boot time which would then be capable of providing accurate coordinate estimates to applications and their overlays on request. In essence, the coordinate system could then be seen as a component of a "virtual infrastructure" that supports a wide range of overlays and applications.

But a system providing an "always-on and large scale coordinate service" would also likely be a prime target for hackers, as its disruption could result in the mis-functioning or the collapse of very many applications. Indeed, as the use of overlays and applications relying on coordinates increases, one could imagine the release of worms and other malicious software whose purpose is to attack the coordinate system. It should also be noted that as current proposals for coordinate systems assume that the nodes partaking in the system cooperate fully and honestly with each other – that is that the information reported by probed nodes is correct – this could also make them quite vulnerable to malicious attacks. In particular, insider attacks executed by (potentially colluding) nodes infiltrating the system could prove very effective. In this paper we study just how potent this danger is for the Vivaldi coordinate system and to the best of our knowledge, this work constitutes the first study identifying some threats and analyzing their effects on network positioning and distance prediction systems. We believe that securing the base of distance prediction for many applications is much more critical, than detailing security of the artifacts of any particular application.

We identify three types of potential attacks against Vivaldi. Specifically, we study how these attacks can lead to inferior application performance due to inaccuracy of prediction. We analyze simple ways that allow malicious nodes to take control of the embedding coordinates system, as they are able to impose positions in the network to other honest nodes, without being detected. We also demonstrate that it is easy to perform Denial of Service (DoS) attacks on such systems. Finally, we study how conspiracy can be achieved in these systems and how much it could affect them. The "effectiveness" of these attacks on the target systems are demonstrated through

extensive simulations.

The rest of the paper is organized as follows. Section 2 provides a brief overview of the Vivaldi embedding coordinates system. We identify and classify the attacks in Section 3. We demonstrate and study the effects of theses attacks, through extensive simulations, in Section 4. Section 5 concludes the paper.

II. Background

In this section, we give a brief survey of recently proposed systems for computing coordinates to network positioning.

A. Fixed Landmark-based coordinate systems

These systems involve a set of landmark nodes, where other nodes compute coordinates according to measurements to these landmarks.

In GNP [6], the coordinates of the landmarks are first computed by minimizing the error between the measured distances and the estimated distances between the landmark nodes. An ordinary host derives its coordinates by minimizing the error between the measured distances and the estimated distances to the landmarks. GNP uses the Simplex Downhill method to compute node coordinates.

Lighthouse [7] is an extension of GNP that is intended to be more scalable. Although it has a special set of landmark nodes, a Lighthouse node that joins, does not have to query those global landmarks. Instead, it can query any existing set of nodes to find its coordinates relative to that set, and then transform those coordinates into coordinates relative to the global landmarks.

NPS [8] builds a hierarchical network positioning system based on GNP, where all nodes could serve as landmarks (Reference points) for other nodes. It uses a decentralized algorithm. Instead of sending measurements to a central node that runs the Simplex algorithm to determine landmark coordinates (as GNP does), each NPS landmark re-runs the minimization itself each time it measures the latency to a new node, called reference point. Any node that has determined its position can be chosen by a membership server to be a reference point. NPS has then been studied in order to reduce the load on fixed landmarks. This system scales therefore to thousands (even millions) of nodes. It includes a strategy for mitigating the effects of (as opposed to colluding) malicious nodes, that could potentially lie about their positions and/or inflate network distances by holding onto probe packets. The basic idea is to eliminate a reference point if it fits poorly in the Euclidean space compared to the other reference points. Each node, when computing its coordinates, based on different reference points measurements, would reject the reference that provides a relative error significantly larger than the median error among all reference nodes. It aims not to be affected by a minority of independent malicious nodes.

B. Decentralized Internet coordinate systems

PIC [9] is one of the recent decentralized coordinate systems using the Simplex Downhill to minimize an objective distance error function (sum of relative errors). It does not require explicitly designated landmarks. It uses an active node discovery protocol to find a set of nearby nodes to use in computing coordinates. Different strategies such as random nodes, closest nodes, and a hybrid of both, are proposed. PIC aims to defend the security of its coordinate system against independent malicious participants using a test based on the triangle inequality. In [10] and [12], it is proven that RTT violations of the triangle inequality are common and persistent. A security based on the fact that the triangle inequality systematically holds, may lead to degrading the system performance when no malicious are inside.

Vivaldi [13], the focus of our present study, 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. Vivaldi defends against high-error nodes, but not malicious nodes. It is described in further details in the following section. Finally, BBS [14] performs a similar simulation to calculate coordinates, simulating an explosion of particles under a force field.

C. Vivaldi Overview

Vivaldi is fully distributed, requiring no fixed network infrastructure and no distinguished nodes. A new node compute its coordinates after collecting latency information from only a few other nodes. Basically, Vivaldi places a spring between each pair of nodes (i, j) with a rest length set to the known RTT (L_{ij}). The current length of the spring is considered to be the distance between the nodes in the coordinate space. The potential energy of such a spring is proportional to the square of the displacement from its rest length: the sum of these energies over all springs is the error function that Vivaldi nodes try to minimize.

An identical Vivaldi procedure runs on every node. Each sample provides information that allows a node to update its coordinates. The algorithm handles high error nodes by computing weights for each received sample. The sample used by each node, i is based on measurement to a node, j, its coordinates x_j and the estimated error reported by j, e_j. A relative error of this sample is then computed as follows:

$$e_s = \mid \parallel x_j - x_i \parallel - RTT_{measured} \mid / RTT_{measured}$$

The node then computes the sample weight balancing local and remote error : $w = e_i/(e_i + e_j)$, where e_i is the node's current (local) error. This sample weight is used to update an adaptive timestep, δ defining the fraction of the way the node is allowed to move toward the perfect position for the current sample: $\delta = C_c \times w$, where C_c is a constant fraction < 1. The node updates its local coordinates as the following:

$$x_i = x_i + \delta \cdot (RTT_{measured} - \parallel x_i - x_j \parallel) \cdot u(x_i - x_j)$$

where $u(x_i - x_j)$ is a unit vector giving the direction of i's displacement. Finally, it updates its local error as $e_i = e_s \times w + e_i \times (1 - w)$. The reader should note that after convergence of a Vivaldi system, the relative local error variation is of the order of a few percent (e.g. +/-0.05).

Vivaldi considers a few possible coordinate spaces that might better capture the underlying structure of the Internet. Coordinates embedding map into different geometric space, where nodes are computing their coordinates, e.g., 2D, 3D or 5D Euclidean spaces, spherical coordinates, etc. Vivaldi also introduces the *Height model*, consisting in an Euclidean coordinate space augmented with a height vector. The Euclidean portion models a high-speed Internet core where latencies are proportional to geographic distance, and the height vector models the time it takes packets to travel the access link from the node to the core. In [13], authors show that the more vectors an Euclidian space has, the more accurate the Vivaldi system is. Moreover, results prove that height vectors perform better than both 2D and 3D Euclidean coordinates.

III. Threats exploiting cooperation and attacks classification

We classify attacks and identified threats that malicious nodes may seek to carry out on a positioning coordinate-based systems. We consider malicious nodes that have access to the same data as a legitimate user. This means that participants are not completely trusted entities, or that malicious have the ability to bypass any authentication mechanisms. Malicious nodes are able to send misleading information when probed, or send manipulated information after receiving a request or affect some metrics observed by chosen targets. The main classes of attacks on positioning system behavior are:

1) Isolation: where nodes would be isolated in the network. The attack could target a particular node, in order to convince the victim that it is positioned in an isolated zone of the network. The final goal of such attacks can be obliging the victim to connect to the malicious node, as the closest node in that zone, in order to perform traffic analysis or packets dropping, man in the middle attacks, etc. One way a malicious node can conduct this attack is to delay probes sent by the victim, and to falsify its proper coordinates, so that the victim's computed coordinates are set to a value large enough, to be far from other nodes and at the same time near the attacker's coordinates.
2) Repulsion: where a malicious would convince its victims that it is positioned far from other participating nodes in order to reduce its attractiveness, and then alleviating its resource consumption by not cooperating in the application progress. Ways to perform such attacks are to make its conditions (performance, position) seem worse than they actually are. This is accomplished by means of lying in responses to active probes or by manipulating the coordinates transmitted to other nodes or to a set of central entities, such as landmarks.
3) Disorder: The main goal here is to create chaos as a form of denial of service (DoS) attack. This results in high errors in the positioning results, or the non-convergence of the algorithm. The attack consists only in maximizing the relative error of nodes in the system, either passively by not cooperating or by falsifying its coordinates or actively by delaying probes.
4) System Control: This attack is possible in case of coordinates-based systems that allow "normal" nodes to be considered as landmarks, i.e. most of the existing systems except the centralized systems. In hierarchical systems for example, such as NPS [8], nodes would try to get higher in the hierarchy in order to fool and influence the maximum number of correct nodes.

The classes of attacks briefly described above can either be carried out by malicious nodes in an independent manner or as a conspiracy created by colluding nodes. Collusion is likely in a scenario where attack propagation happens through the now well tested means used in today's DDoS attacks (e.g. worms, etc).

It should be noted that all attacks, be they explicitly aimed at disrupting the whole system or skew the coordinates of a single node will all result in some distortion of the coordinate space. This is because of the cooperation between the nodes that will act as a catalyzer to the propagation of errors to other (non directly targeted) nodes.

Finally, some security mechanisms in coordinates-based systems are used to try and filter out malicious nodes, although these are still rather primitive and still in their infancy and definitely cannot defend against all types of attacks. To disrupt the system in such cases, malicious nodes can fool other well behaved nodes by turning the system against itself. As these security systems often refer to a median error to filter high error nodes, for instance, a malicious node can switch between correct information and biased ones, and at the same time inject traffic to influence the performance of other nodes. This would affect the measurements a correct node is reporting for example. The ultimate goal of such attacks is to consider correct nodes as malicious, and vis versa.

IV. Performance evaluation

A. Performance Indicators

We use the relative error (defined in II-C) as our main performance indicator. We compute the average relative error over all nodes to represent the accuracy of the overall system. Since our focus is on measuring the impact of malicious nodes on the system, we also introduce the *relative error ratio* (called Ratio), which is the relative error measured in presence of malicious nodes normalized to the performance of Vivaldi without cheats used as a best case reference (i.e. $\mathrm{error\_ratio} = \mathrm{error}/\mathrm{error}_{\mathrm{ref}}$). Obviously, a value for the error ratio above 1 indicates a degradation in accuracy.

As a worst case reference point, we also compute the relative error of a coordinate system where nodes choose their coordinate at random. In this random scenario, all nodes choose their coordinate randomly in the interval $[-50000, 50000]$.

B. Simulation set up

In this section, we present the results of an extensive simulation study of attacks against the Vivaldi system. For the simulation scenarios, we used the p2psim discrete-event

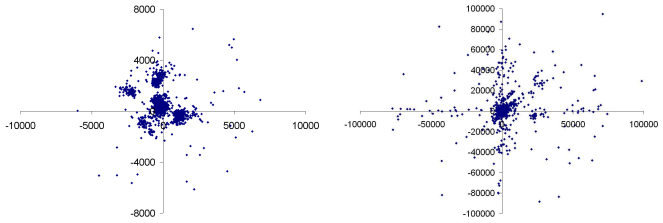
simulator [15], which comes with an implementation of the Vivaldi system. We also used the "King" data set to model Internet latency based on real world measurements. This dataset contains the pair-wise RTTs between 1740 Internet DNS servers collected using the King method [16]. This was used to generate a topology with 1740 overlay nodes, from which we derived various group sizes by picking nodes at random (unless otherwise stated, the group consists of all the 1740 nodes). Each scenario was repeated 10 times with the malicious nodes selected at random within the group. We consider groups with 10%, 20%, 30%, 40%, 50% and 75% of malicious. In view of the infection rates of recent worm epidemics, we believe these values to be realistic, both during and for a long time after an outbreak.

Each Vivaldi node has 64 neighbours (i.e. is attached to 64 springs), 32 of which being chosen to be closer than 50 ms. The constant fraction C_c for the adaptive timestep (see section II-C) was set to 0.25. These values are those recommended in [13]. The system was considered stabilized when all relative errors converged to a value varying by at most 0.02 for 10 simulation ticks. We observed that Vivaldi without malicious nodes always converged within 1800 simulation ticks, which represents a convergence time of over 8 hours (1 tick is roughly 17 seconds). Unless otherwise stated, our results are obtained for a 2-dimensional coordinate space.

Finally, we also considered the different attacks in a "genesis" or "injection" context. In a genesis attack, the malicious nodes are present from the system's creation time, while in an injection attack the malicious nodes are introduced in a system that has already converged. This distinction allows for a better understanding of the impact of the attacks on convergence and stability.

C. Attack methods in action on Vivaldi

1) Disorder Attack: We first discuss ways to achieve Disorder attacks in Vivaldi. As it is a fully-distributed algorithm relying on cooperation of nodes in order to ensure accuracy of the computed coordinates, it seems easy to fool honest nodes. The disorder attack has no specific objective, but false coordinates computations and high effective error. When requested, a malicious node would send a randomly selected coordinate x_j, associated with a very low error, e_j = 0.01. Moreover, each node's measurement is delayed by a randomly generated value in [100..1000] ms. In this first scenario, it is not necessary to care about lie consistency, as Vivaldi is based on error sent by the requested node to adjust its adaptive timestep. Even if the measured distance $RTT_{measured}$ is not consistent with the sent coordinates x_j, the victim i would consider itself as a high error node, and would try to adjust its coordinates by a great adaptive timestep value, due to the fact that j sends a low error. Figure 1 illustrates the effects of malicious nodes on the coordinate space of Vivaldi. we can see that the topology we used exhibits a clear "cluster" structure that disappear in the presence of only 10% of malicious nodes. This is because, in this disorder attack, the attackers keep "jolting" the system and the errors introduced "ripple" through the system, propagated through normal operations of the honest nodes.



(a) Vivaldi (b) Vivaldi with 10% malicious

Fig. 1. *Effect of Disorder attack with 1740 nodes*

Figure 2 depicts the relative error variation in function of time, for our full set of 1740 nodes, representative of the impact of the malicious nodes on system convergence. With no malicious nodes, Vivaldi was observed to stabilize around 1500 simulation ticks. Note that in the first part of the simulation (0-1500 ticks), the ratio metrics increase can be due to both the relative error of the reference Vivaldi decreasing to its converged value and the relative error caused by the malicious nodes increasing. It is interesting to note that, in the presence of enough malicious nodes, despite the system converging in the sense that the relative errors at each node stabilize, these errors are so high that a great variation of the coordinates of a node barely affects the associated error. In other words, the coordinates of the nodes keep showing great variations and do not stabilize but the error introduced by such constant movement is stable because there is already so much chaos in the system. In essence, the system is deemed to converge because it doesn't get any better nor any worse.

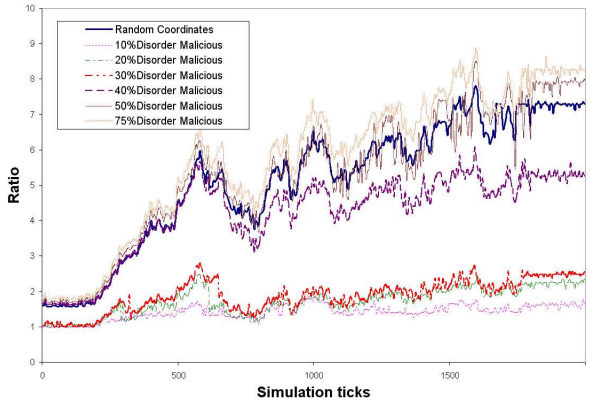


Fig. 2. Genesis Disorder Attack: Variation of the ratio of average relative error in presence of malicious nodes.

Figure 3 shows the cumulative distribution of the relative error of the victims of a genesis disorder attack. We clearly see that from 30% of malicious nodes the impact on the system can be considered as very serious with many nodes seeing a large increase in their relative errors. For a proportion of 50% or more malicious nodes, the system collapses with over half of the honest nodes computing coordinates that are similar or

worse than if chosen randomly.

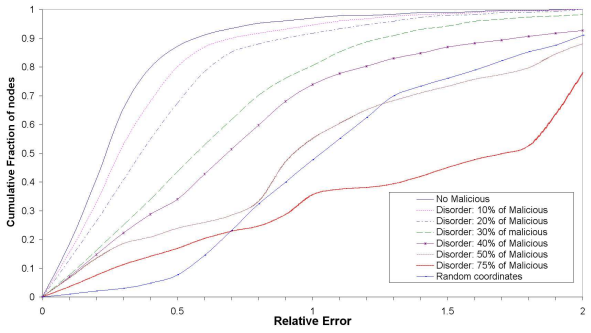


Fig. 3. Genesis disorder attack: CDF of relative error at simulation tick = 2000

Figure 4 represents the impact of the space dimension on the attack. In this figure, the average relative error of honest nodes is measured after convergence. We see that the most accurate the Vivaldi system is in the absence of malicious nodes, the most vulnerable it is to the disorder attack. This is because the variation of more coordinates components for a point in a larger space results in higher displacement in that space. This observation is compounded for the 2-dimensional space augmented by a height as a variation of the height yields a greater effect on the node displacement. We also observe that in all cases, Vivaldi with half the population of malicious nodes is worse than a random coordinate system.

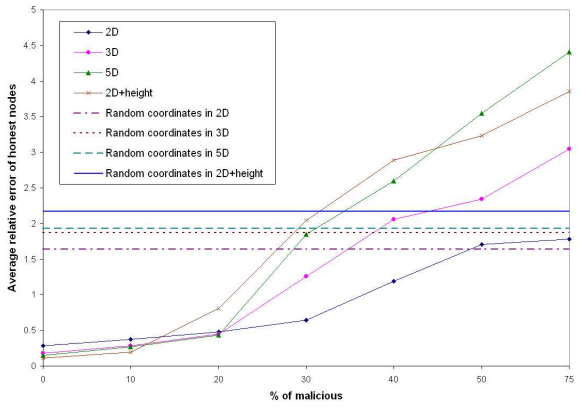


Fig. 4. Genesis Disorder Attack: Impact of space dimensions

The effect of an injection disorder attack, where malicious nodes appear only after the system has converged, are depicted in figures 5 and 6. Figure 5 shows the evolution in time of the average relative error ratio, clearly showing that enough attackers can not only quickly destabilize a converged system but prevent re-convergence. Figure 6 shows the impact of the attack as a function of the system size as measured a long time after the attack started. We see that a larger system is more difficult to impact for a same proportion of attackers. This is consistent with the fact that a larger Vivaldi system is more accurate, but also establishes that Vivaldi finds increased strength in a larger group. Put simply, this is because as one increases the number of springs in the system, the energy needed to disrupt it is higher. In our case, a larger group means more "good" forces to counteract and dissipate the effect of the malicious ones.

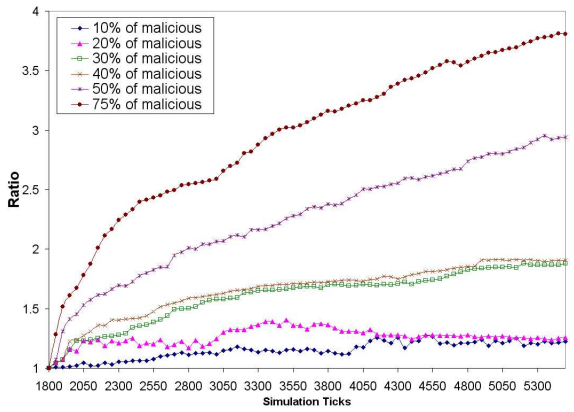


Fig. 5. Injection of Disorder Attackers: average relative error ratio.

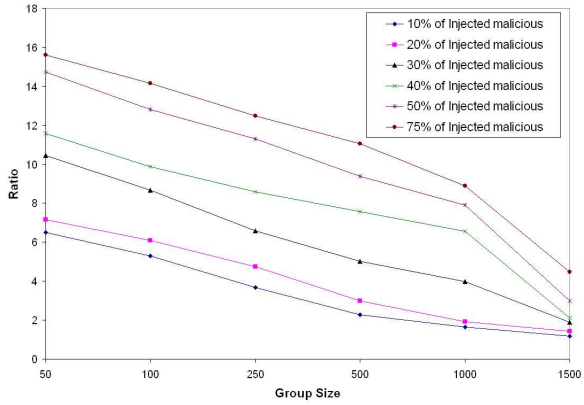


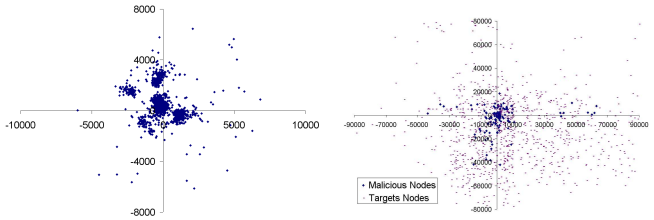
Fig. 6. Injection of Disorder Attackers: Impact of system size on the attack.

2) Repulsion Attack: In this scenario, malicious nodes are trying to isolate some nodes in the network. The first attack consists in fixing coordinates where to isolate all requesting nodes, say X_{target}. It is important to notice that this value is set high enough to allow lie consistency. This means that the predicted distance after the lie should be equal to the measured distance. In fact, since we assume that a malicious node cannot shorten a distance measurement, but can however delay it, we must set the coordinates of both the victim and the malicious node to be consistent with this fact. Although for most network positioning systems, application probes are used, for generality purposes we design and test the attacks assuming ICMP ping probes. We assume here that malicious nodes know the current coordinates of their targets, $X_{Current}$, by means of previous requests for example. Malicious nodes are then able to compute the needed RTT that are consistent with the lie,

$$RTT = (\parallel X_{target} - X_{Current} \parallel / \delta) + \parallel X_{target} - X_{Current} \parallel$$

and to delay the measured RTT by: $RTT_{needed} - 2 \cdot (ReceivedTimestamp - SendTimestamp)$. Each malicious node is selecting a random coordinate that is far away from the origin.

The effect of this attack can be visualized in figure 7, for our full-size system and after convergence of the normal Vivaldi system. In this version of the attack, each malicious node sets a target coordinate independently for every honest node.



(a) Vivaldi (b) Vivaldi with 10% malicious

Fig. 7. *Effect of Repulsion attack with 1740 nodes*

Figure 8 shows the cumulative distribution function of the measured average relative error after convergence in a genesis repulsion attack. The gentler slope of the curves indicates that the impact of this type of attack is greater than in the case of a disorder attack (see fig. 3). This is because a repulsion attack is more structured and more consistent than a disorder attack, since the chosen target coordinate is always the same for every victim-attacker pair.

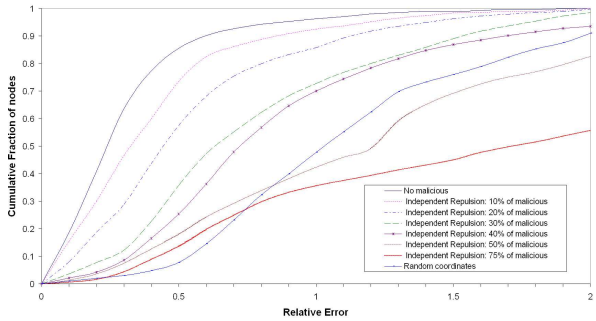


Fig. 8. CDF of relative error.

We study the effect of space dimension on the attack in figure 9. Again, the results confirm that the more accurate the system is without malicious nodes, the more vulnerable it is to attacks, which highlights a fundamental trade-off between accuracy and vulnerability.

So far, the repulsion attack consisted each attacker attacking every other node. Figure 10 shows the effect of a modified repulsion attack where each attacker independently attacks a subset of the other nodes. Each attacker chooses its own target subset independently of the other, along with their target coordinate value. However, the target subset size is fixed and equal for all attackers. We see that small subsets chosen independently result in a less effective attack and that target subsets smaller than half the overall population,

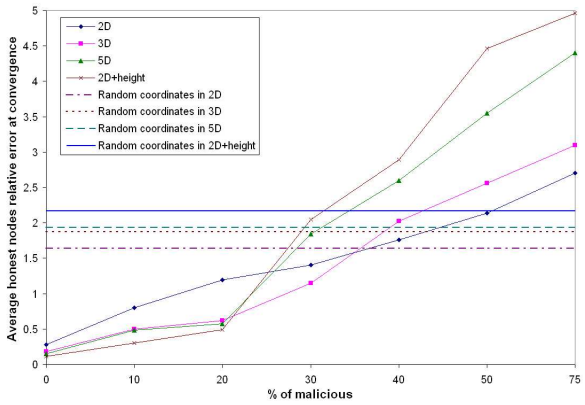


Fig. 9. Genesis Repulsion Attack: impact of space dimensions.

there is no great difference in effectiveness when the set of attackers constitutes less than half the population. This can be explained by the fact that in such conditions the attack gets "diluted", giving the system plenty of opportunity to correct itself through nodes that are under no, or very little, attack.

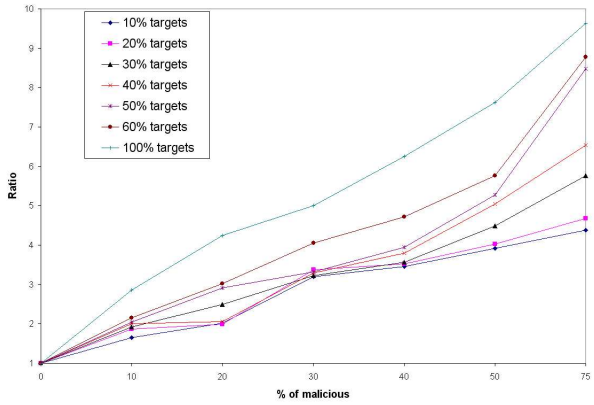


Fig. 10. Genesis Repulsion Attack on subsets of target nodes.

Figure 11 shows the response of a system under injection repulsion attack as a function of system size. As in the case of a disorder attack, larger systems reduce the impact of the attack. However, because a repulsion attack is much more consistent than a disorder attack, the system is less effective at countering the effects. This is why we observe higher values for the average relative error and a much gentler slope of the curve than in figure 6.

3) Colluding Isolation attack: This is a repulsion attack where the attacker behaves consistently in a collective way. They could, for instance, try and move all honest nodes consistently away from a same designated target node. That is, they agree on a distance from the chosen node for each victim and collectively and consistently direct victims towards their designated coordinate. Figure 12 shows the effects for such an attack.

Figure 13 depicts the effects of a genesis colluding isolation attack on the convergence properties of the system. The salient

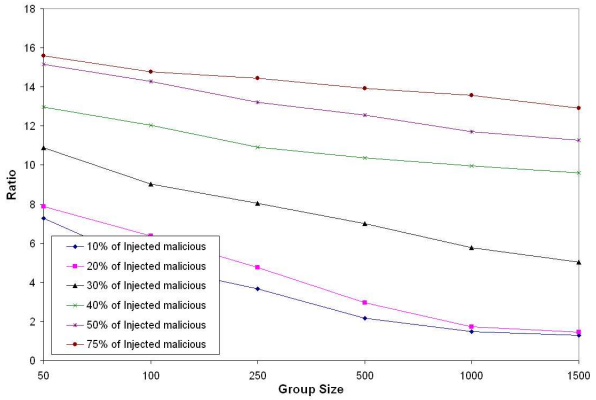
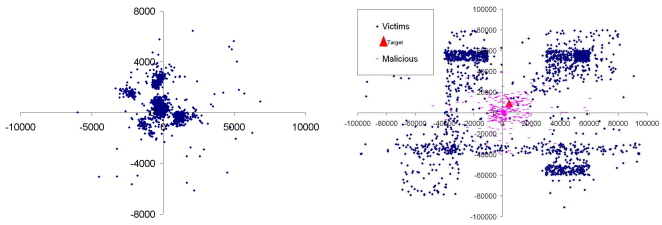


Fig. 11. Injection Repulsion Attack: effect of system size



(a) Vivaldi (b) Vivaldi with 10% malicious

Fig. 12. *Effect of Colluding Isolation attack with 1740 nodes*

result is that the system can quickly become worse than a random coordinate system. Indeed, from 30% of malicious nodes in the system, the accuracy becomes equal or worse than if nodes chose their coordinates at random. This clearly demonstrates that colluding attacks are very potent due to their better structure and can have a great adverse impact on overall system performance.

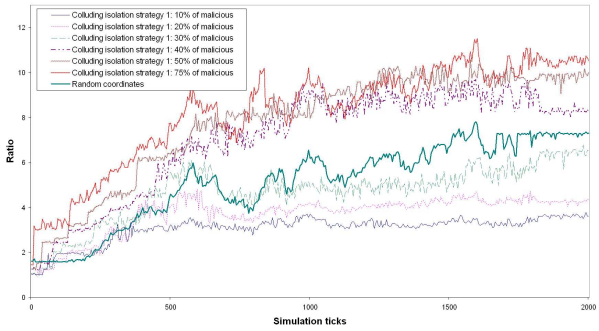


Fig. 13. Genesis Colluding isolation Attack: average relative error ratio

Another type of colluding isolation attack is for the attackers to set their coordinates in a remote area of the coordinate space (so that they are clustered in that area) and then to choose a victim target node and convince it that its own coordinate are within the attacker cluster. The target coordinate is set before the attack begins and agreed by all attackers.

We observe in figure 14 the variation of the relative error of the target through time. We see that the first type of colluding

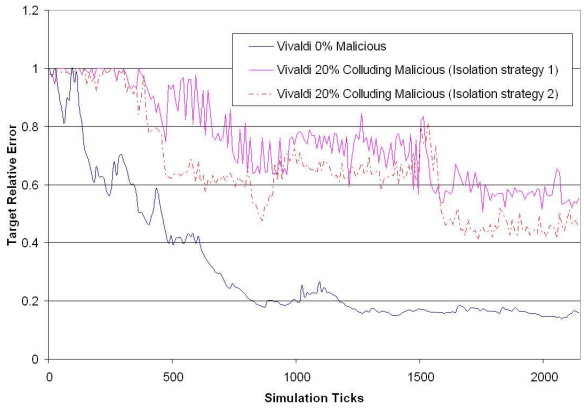


Fig. 14. Colluding Isolation attack 2: target relative error

isolating attack (consisting in repelling all other honest nodes from a chosen target) is more effective than trying to lure a target into a remote area of the space. Intuitively, this is because much more error is introduced in the system when more nodes are pushed away from their correct position, thus resulting in more distortion of the coordinate space with greater repercussion on the final position of the target node. This is indeed confirmed by the results of figure 15 that depicts the cumulative relative error for the nodes in the system under both types of colluding attacks.

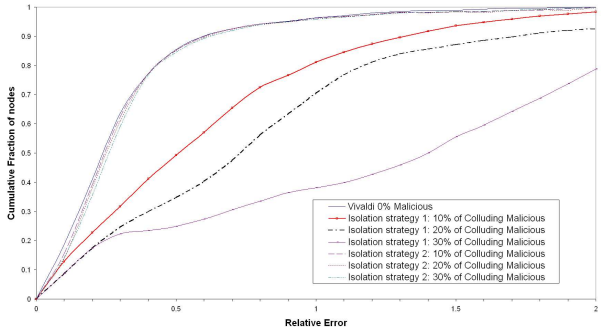


Fig. 15. Colluding Isolation Attack: CDF of relative errors.

4) Combined attacks: In the context of system offering an always-on and large scale coordinate service, it is plausible to assume a constant and permanent low level at malicious nodes. Indeed, in the previous sections we have examined the effects of attack outbreaks. But in the wild, as has already been observed after major worm outbreaks and security warning, once an outbreak has been contained and resolved, one can expect that some small portion of the systems are not upgraded for a very long time after the release of the necessary patches. This is especially true in the case of systems that are under many different administrative controls (as is the case for home personal computers). Figure 16 shows the impact of such low level of combined attacks on Vivaldi, where colluding nodes implement strategy 1 of the colluding isolation attack. In these combined attacks, the percentage of malicious nodes of each

type is the same. This figure shows that fairly low level of malicious nodes can still have a sizeable impact on the overall system performance, which, in turn, indicates that return to normality after an attack may take an extremely long time, if at all possible.

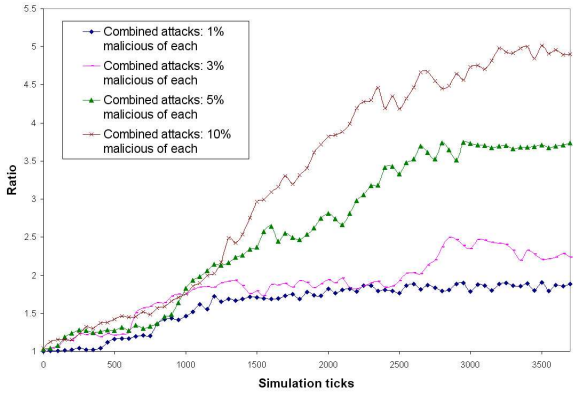


Fig. 16. Combining attacks: impact on convergence.

Finally, figure 17 confirms that larger systems are more resilient and recover better than smaller ones.

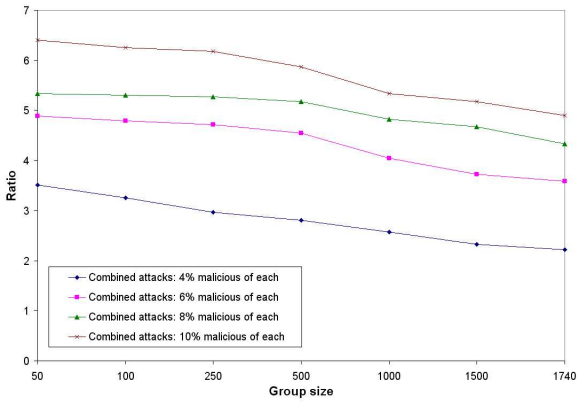


Fig. 17. Combined attacks: effect of system size.

V. Conclusion

In this paper, we have studied various types of attacks on the Vivaldi coordinate system. One of our salient findings is that larger systems are consistently more resilient than smaller ones. Given the observation in [13] that larger systems are more accurate and the well known fact that larger system converge slower at start-up time, there seems to be a compelling case for large-scale coordinate systems to be built as a virtual infrastructure service component. The paradox is of course that always-on, large scale systems supporting many different applications will always attract more attacks than systems with a smaller reach, while the large size of the system itself would act as a particularly good terrain to create especially virulent propagation of the attack.

Our results also show that there is an intrinsic trade-off to be made between accuracy and vulnerability. Indeed, we have shown that the more accurate the system for a given system size, the more susceptible it was to a same proportionate level of attack.

Also, we have shown that while an attack is in full swing, the performance of the Vivaldi coordinate system (and of the applications it supports) can easily degrade below that of a system where coordinates are chosen randomly, whilst the aftermath of an attack could have very long lasting effects on the system due to a small number of remaining malicious nodes.

In our future work, we will study the impact of malicious attacks against other families of coordinate systems, including those that incorporate some defenses against attacks.

References

[1] A. Rowstron and P. Druschel, *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*, in Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms, Heidelberg, Germany, November, 2001.

[2] J. Kubiatowicz et al., *OceanStore: An Architecture for Global-Scale Persistent Storage*, in Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), Cambridge, November 2000.

[3] Y. h. Chu, S. G. Rao and H. Zhang, *A case for end system multicast*, In Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), Santa Clara, June 2000.

[4] `www.skype.com`

[5] S. Rewaskar and J. Kaur, *Testing the Scalability of Overlay Routing Infrastructures*, in Proceedings of of the Passive Active Measurement (PAM) Workshop, Sophia Antipolis, France, April 2004.

[6] T. E. Ng, and H. Zhang, *Predicting internet network distance with coordinates-based approaches*, in Proceedings of the IEEE INFOCOM, New York, June 2002.

[7] M. Pias, et al., *Lighthouses for Scalable Distributed Location*, in Proceedings of International Workshop on Peer-to-Peer Systems (IPTPS0, Berkeley, February 2003.

[8] T. E. Ng and H. Zhang, *A Network Positioning System for the Internet*, in Proceedings of the USENIX annual technical conference, Boston, June 2004.

[9] M. Costa, et al., *Practical Internet coordinates for distance estimation*, in Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS), Tokyo, March 2004.

[10] E. K. Lua et al., *On the accuracy of Embeddings for Internet Coordinate Systems*, in Proceedings of Internet Measurement Conference (IMC), Berkeley, October 2005.

[11] A. Walters, K. Bauer and C. Nita-Rotaru, *Towards Robust Overlay Networks: Enhancing Adaptivity with Byzantine resilience*, Technical Report CSD TR 05-026.

[12] H. Zheng et al., *Internet Routing Policies and Roun-Trip Times*. In Proceedings of the Passive Active Measurement (PAM), Boston, March 2005.

[13] F. Dabek, R. Cox, F. Kaashoek and R. Morris, *Vivaldi: A decentralized network coordinate system*. In Proceedings of the ACM SIGCOMM, Portland, Oregon, August 2004.

[14] Y. Shavitt and T. Tankel, *Big-bang simulation for embedding network distances in euclidean space*, in Proceedings of the IEEE INFOCOM, San Francisco, April 2003.

[15] A simulator for peer-to-peer protocols. `http://www.pdos.lcs.mit.edu/p2psim/index.html`

[16] K. P. Gummadi, S. Saroiu, and S. D. Gribble, *King: Estimating Latency between Arbitrary Internet End Hosts*, in Proceedings of SIGCOMM Internet Mesasurement Workshop (IMW), Pittsburgh November 2002.