PhD-FSTC-2012-28
Faculty of Science, Technology and Communication

Ecole doctorale IAEM Lorraine

DISSERTATION

Defense held on 16/10/2012 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

AND

DOCTEUR DE L'UNIVERSITÉ DE LORRAINE

SPÉCIALITÉ: INFORMATIQUE

by

Sheila BECKER
born on 11 May 1983 in Luxembourg

Conceptual Approaches for Securing Networks and Systems

Acknowledgements

This thesis would not have been written without all the great encouragement and help I was blessed to obtain throughout the years. Therefore, I would like to thank first of all Prof. Dr. Thomas Engel not only for welcoming me in his research group Secan-Lab at the University of Luxembourg but also for encouraging me to continuing my studies, if not for his support and enthusiasm, my research interests and curiosity would never have been aroused. Also, I would like to thank Dr. Hab. Olivier Festor for welcoming me in the research group Madynes at INRIA-LORIA Nancy and for the great encouragements and cooperation during the past years. Special thanks belong to Dr. Hab. Radu State, who always showed me that if there is a will a way will be found and who did not get tired in encouraging me, even during times when I did not see light at the end of the tunnel. I also would like to thank Prof. Dr. Cristina Nita-Rotaru for the fruitful cooperation and for welcoming me in the research group Dependable and Secure Distributed Systems Lab (DS2) at Purdue University for nine months as a visiting researcher. For the same cooperation I would also like to thank Jeff Seibert and David Zage. Furthermore, I would like to thank Dr. Hab. Ludovic Mé and Prof. Dr. Burkhard Stiller for accepting the reviewer positions of my thesis.

Special thanks belong to Fulbright[1] that allowed me to experience the life at the Purdue University and to the students of Purdue Fulbright Association who made my stay full of wonderful memories.

We all know, that a working environment is only as nice as the relations between the group-members. Therefore, I would like to thank all the group members of Secan-Lab, Madynes, and DS2 for making a fruitful, helpful, and funny working environment. Nevertheless, special thanks need to be mentioned, and in this case I want to thank Magali Martin and Nathanaëlle Minard for being always very helpful especially for administrative issues and for not missing a single opportunity to put a smile on my face. The daily work would have been much less interesting if I would not have been able to share the office with Cynthia Wagner, who not only had an open ear for my professional and private concerns but also always knew how to make me feel better. I will miss our chin wag.

Furthermore, I would like to thank my family for their support. I am especially thankful for the love, the understanding and the support of my parents, Madeleine and Carlo, who always believed in me. They gave me strengths on weak days and showed me the sun on rainy days. I also received a lot of support and comfort from my friends. Tania Kremer, Cynthia Wagner, Tom Leclerc and Hyojeong Lee, I am deeply grateful for all your patience, love, and encouragement.

[1]http://www.iie.org/fulbright

Abstract

Peer-to-peer real-time communication and media streaming applications optimize their performance by using application-level topology estimation services such as *virtual coordinate systems.* Virtual coordinate systems allow nodes in a peer-to-peer network to accurately predict latency between arbitrary nodes without the need of performing extensive measurements. However, systems that leverage virtual coordinates as supporting building blocks, are prone to attacks conducted by compromised nodes that aim at disrupting, eavesdropping, or mangling with the underlying communications.

Recent research proposed techniques to mitigate basic attacks (*inflation, deflation, oscillation*) considering a single attack strategy model where attackers perform only one type of attack. In this work, we define and use a game theory framework in order to identify the best attack and defense strategies assuming that the attacker is aware of the defense mechanisms. Our approach leverages concepts derived from the Nash equilibrium to model more powerful adversaries. We apply the game theory framework to demonstrate the impact and efficiency of these attack and defense strategies using a well-known virtual coordinate system and real-life Internet data sets.

Thereafter, we explore supervised machine learning techniques to mitigate more subtle yet highly effective attacks (*frog-boiling, network-partition*) that are able to bypass existing defenses. We evaluate our techniques on the Vivaldi system against a more complex attack strategy model, where attackers perform sequences of all known attacks against virtual coordinate systems, using both simulations and Internet deployments.

Contents

List of Figures

List of Tables

Chapter 1

Résumé en français

1.1 Introduction

Actuellement, l'aspect le plus important pour la conception des réseaux est le passage à l'échelle, car les réseaux sont composés de plus en plus de différentes parties, qui doivent supporter un grand nombre d'utilisateurs. En conséquence les architectures des réseaux client/serveur sont de plus en plus substituées par les réseaux pair-à-pair (P2P).

Les services de partage de fichiers (par ex.: BitTorrent) ont aidé les réseaux pair-à-pair à atteindre une grande popularité. Au fur et à mesure, ces services de partage de fichiers étaient de plus en plus utilisés et pour améliorer l'efficacité des réseaux pair-à-pair, des systèmes de coordonnées virtuelles ont été inventés. L'idée principale de ces systèmes est que les nœuds sont associés à une position dans un espace virtuel et les distances entre les nœuds avoisinants représentent la relation de la latence entre ces nœuds. Ces systèmes prennent les latences pour positionner les nœuds dans un espace virtuel.

Pourtant ces systèmes sont la cible d'attaques ayant pour but de diminuer la précision et les performances de ces systèmes. Spécifiquement, il existe des attaques lentes et subtiles, que les mécanismes de défenses existants ne peuvent pas détecter. Ceci car ces attaques restent en dessous des seuils de détection et sont si lentes qu'elles sont considérées comme un comportement bienveillant.

Dans une première étape, nous analysons les différentes stratégies d'attaques et de défenses d'une manière systématique pour en déduire des stratégies optimales. Avec ces analyses nous étendons les validations des mécanismes de défenses existants, jusqu'à présent seulement faites à base des expérimentations.

Ensuite, nous développons une nouvelle méthode de détection qui utilise l'apprentissage automatique. Notre méthode peut détecter les attaques simples ainsi que les attaques subtiles et lentes. Nous comparons trois algorithmes d'apprentissage supervisé différents. De plus, nous montrons que notre méthode peut également détecter des stratégies d'attaques plus complexes.

Ce chapitre a la structure suivante : Premièrement, nous expliquons l'état de l'art des systèmes aux coordonnées virtuelles, ainsi que l'état de l'art des attaques contre ces systèmes et leurs mécanismes de défenses dans la section 1.2. Deuxièmement, nous illustrons nos contributions (voir section 1.3). Finalement nous terminons avec la conclusion et les travaux futurs dans la section 1.4.

1.2 Etat de l'art

Beaucoup de recherche était nécessaire afin de développer des méthodes pour mieux organiser les réseaux pair-à-pair. Enfin, les systèmes aux coordonnées virtuelles ont été développés. Pour prévoir les distances entre les nœuds, ces systèmes ne considèrent pas des mesures additionnelles pour calculer la latence pour ne pas charger le réseau, mais ils estiment les distances à partir de messages existants.

Il existe deux types de systèmes aux coordonnées virtuelles. Le premier type a besoin de points de repère pour être capable de localiser les nœuds. Le deuxième type est complètement décentralisé et n'a pas besoin de points de repère.

Les systèmes aux points de repère dépendent des composants infrastructurels, par ex. : un ensemble de serveurs, pour prévoir les distances entre les nœuds. Ces points de repère sont soit prédéfinis [40, 75, 99], soit choisis aléatoirement [76, 80].

1.2.1 Systèmes aux coordonnées virtuelles décentralisés

A cause des points de repère, ces systèmes ont des difficultés à passer à l'échelle et pour vaincre ce problème, les systèmes décentralisés ont été proposés. L'objectif des systèmes aux coordonnées virtuelles décentralisés [30, 32, 62, 63] est de produire et maintenir efficacement des ensembles de coordonnées, qui prédisent soigneusement les latences entre les nœuds.

Les approches décentralisées utilisent un ensemble de référence composé des nœuds avoisinés. Un nœud calcule les distances par rapport à cet ensemble, car ce n'est pas possible de passer à l'échelle quand un nœud calculera ses distances par rapport à tous les nœuds. Les auteurs de PIC [30] ont démontré une efficacité maximale si les nœuds de l'ensemble avoisiné sont choisis d'une moitié aléatoirement et de l'autre moitié entre des nœuds proches physiquement, c.-à-d. des nœuds avec des petites latences. Les approches proposées diffèrent en ce qui concerne la grandeur de cet ensemble, PIC [30] utilise 32 nœuds, PCoord [62, 63] utilise 10 nœuds et Vivaldi [32] 64 nœuds. En outre, les trois approches implémentent des différentes méthodes pour bien estimer les latences entre les nœuds. PIC et PCoord implémentent la méthode Simplex Downhill [71]. Pourtant, Vivaldi minimise la racine de l'erreur de prédiction.

1.2.2 Vivaldi

Dans ce travail, nous nous concentrons sur Vivaldi, comme c'est le système le plus utilisé et le fonctionnement est assez simple pour comprendre et visualiser. En plus, dans [32] c'est montré que ce système produit seulement une petite erreur d'intégration.

En Vivaldi, tous les nœuds reccoivent des coordonnées synthétiques dans un espace euclidien multidimensionnel. Ces coordonnées représentent les latences entre les nœuds, c.-à-d. le temps d'aller-retour entre les nœuds. Pour en recevoir une bonne estimation de la latence, Vivaldi minimise l'erreur de prédiction $Erreur_{pred}$:

$Erreur_{pred} = |L_{act}$ - $L_{est}|$

L_{act} est la latence mesurée entre deux nœuds et L_{est} est la latence estimée, donc la latence calculée à partir des coordonnées de l'espace virtuel.

En fait, l'idée derrière Vivaldi est de décrire un problème de relaxation d'un ressort dérivé du domaine de physique. On peut s'imaginer que deux nœuds sont connectés par un ressort, la longueur actuelle du ressort constitue la latence estimée L_{est}. Quand il existe une tension sur le ressort, les nœuds rectifient leurs positions pour minimiser la tension et en même temps l'erreur de prédiction. Cette procédure de minimisation est appliquée entre un nœud et tous ses nœuds avoisinés. Les nœuds échangent les informations sur la latence régulièrement et à partir de ces informations les nœuds modifient leurs coordonnées pour en trouver l'emplacement qui décrit précisément la latence réelle entre les nœuds.

1.2.3 Attaques sur Vivaldi

Le mode de fonctionnement de Vivaldi assume que les nœuds déclarent des latences et coordonnées correctes, mais un attaquant peut en réduire la performance de Vivaldi en mentant. Quand la performance est réduite, utiliser des systèmes aux coordonnées virtuelles ne rapporte plus une efficacité améliorée pour les systèmes pair-à-pair.

Un attaquant a plusieurs options pour réduire la performance [22, 23, 110, 111]:

L'attaque à l'inflation: l'attaquant peut repousser un nœud victime loin de sa position correcte en signalant des coordonnées similaires aux coordonnées du nœud victime, pour que le nœud victime suppose que le nœud attaquant soit proche de lui, mais il signale une très grande latence. Ce qui a comme conséquence que le nœud victime adapte ses coordonnées à une position loin de sa position correcte.

L'attaque à déflation: L'objectif de cette attaque est de rendre le nœud victime immobile. L'attaquant signale alors soit des coordonnées lesquelles font correspondre la latence estimée avec la latence mesurée du nœud victime, soit une latence modifiée qui correspond à la latence estimée. Par cette attaque, le nœud victime ne peut pas trouver sa position correcte.

L'attaque à l'oscillation: Ici, l'attaquant reporte des mauvaises coordonnées, latences, et erreurs locales, pour empêcher que les autres nœuds puissent converger vers une position correcte. L'objectif est de rendre le système complètement chaotique.

L'attaque à l'inflation-lente: Le principe de cette attaque est le même que pour l'attaque à l'inflation, sauf que cette attaque est beaucoup plus lente pour ne pas risquer d'être détectée par des mécanismes de défenses.

L'attaque à dégroupage de réseaux: Cette attaque est une extension de l'attaque à l'inflation-lente. L'idée est qu'un groupe de nœuds malveillants repousse une partie des nœuds du réseau vers une direction et en même temps un autre groupe de nœuds malveillants repousse l'autre partie du réseau vers la direction opposée.

1.2.4 Mécanismes de défenses

Il y a deux types de défense différents, le premier type est complètement décentralisé [110, 111], le deuxième type par contre utilise un ensemble de nœuds de confiance, comme dans [54], où un filtre

Kalman est utilisé pour modéliser le comportement des nœuds honnêtes à base de l'ensemble de nœuds de confiance. On peut dire que ce deuxième type est à base des points de repère. Avoir besoin des points de repère pour des mécanismes de défenses, résulte en faire des fortes hypothèses par ex. on ne peut pas garantir que ces nœuds de confiance ne soient pas malveillants. A cause de cet inconvénient, nous considérons les mécanismes complètement décentralisés, car ces mécanismes n'ont pas besoin d'hypothèses. Ces mécanismes fonctionnent de la manière suivante [110, 111]:

Détection des valeurs aberrantes spatiales: Ce mécanisme vérifie si les valeurs reportées par un nœud correspondent avec ce qui reportent les autres nœuds avoisinés. Le nœud a une queue avec les valeurs reportées par les autres nœuds avoisinés. La distance Mahalonobis est calculée à partir de toutes ces valeurs, et les valeurs reportées par le nœud en question ne devront pas dépasser un seuil spatial fixe pour que ces valeurs soient acceptées.

Détection des valeurs aberrantes temporelles: Ce mécanisme vérifie si les valeurs reportées par un nœud correspondent avec ce que ce nœud a reporté dans le passé, si les nouvelles valeurs dépassent un seuil temporel, les valeurs ne sont pas acceptées.

Détection des valeurs aberrantes spatio-temporelles: Ici, les mécanismes décrits auparavant sont combinés et les valeurs reportées par un nœud ne doivent ni dépasser le seuil temporel, ni le seuil spatial pour que les valeurs soient acceptées.

Ces mécanismes sont montrés d'être capables de détecter les attaques à l'inflation, déflation et l'oscillation, mais Chan-Tin et al [22] ont montré que ces mécanismes ne détectent ni les attaques à l'inflation-lente ni les attaques à dégroupage de réseau.

1.3 Contributions

Nous avons fait deux contributions principales. Dans la première contribution, nous analysons les stratégies de l'attaquant ainsi que les stratégies des nœuds victimes. Tous les travaux précédents ont validé et analysé les mécanismes de défenses seulement à l'aide des expérimentations. Aucun de ces travaux n'a analysé le fait qu'un attaquant peut avoir des informations sur les mécanismes de défenses, mais la réalité a montré que bien souvent les attaquants ont des connaissances détaillées du système sous attaques.

La deuxième contribution propose un nouveau mécanisme de défense qui est capable de détecter des attaques plus lentes et subtiles, comme l'attaque à l'inflation-lente et l'attaque de dégroupage de réseau. Notre mécanisme utilise l'apprentissage automatique pour discerner le comportement bienveillant du comportement malveillant.

1.3.1 Analyser des stratégies pour sécuriser les systèmes aux coordonnées virtuelles.

Dans ce travail, nous proposons un modèle basé sur la théorie des jeux pour analyser et évaluer les stratégies possibles du défenseur et de l'attaquant dans un système aux coordonnées virtuelles. Nous dérivons des stratégies de défense optimales ainsi que des stratégies d'attaques optimales à l'aide de l'équilibre de Nash. D'après l'équilibre de Nash, nous supposons que les deux adversaires sont rationnels, c.-à-d. qu'ils veulent maximiser leur bénéfice respectif.

Par ailleurs, nous développons une méthode pour éliminer le composant critique des mécanismes de défense précédents: le seuil fixe. Nous utilisons un système à régulation automatique pour avoir une sélection adaptive du seuil.

La structure de cette contribution est la suivante: Premièrement, nous donnons une idée sur la théorie des jeux et l'équilibre utilisé pour en savoir quelles stratégies sont optimales. Deuxièmement, nous établissons notre modèle de jeux.

Théorie des jeux

Nous élaborons des concepts de théorie des jeux afin de dériver les meilleures stratégies pour le défenseur et l'attaquant. Ces stratégies sont obtenues en déterminant l'équilibre de Nash de notre jeu. L'équilibre de Nash [70] est une approche qui consiste à spécifier des choix de stratégie optimales pour tous les joueurs, en considérant qu'aucun des joueurs n'a intérêt à diverger de l'équilibre de Nash, un joueur ne pourrait pas gagner un plus grand profit en choisissant une autre stratégie. Pour calculer l'équilibre de Nash nous utiliserons:

- N : ensemble de n joueurs
- A_i : ensemble de stratégies $(a_i \epsilon A_i)$
- R_i : fonction de gain $\mathrm{A} \rightarrow \mathbb{R}$, où $A = A_1 \times ... \times A_n$

L'équilibre de Nash peut définir une stratégie pure, qui donne une définition complète du comportement du joueur, ou une stratégie mixte, qui est une sélection sur un ensemble de stratégies pures. Une stratégie mixte prévue pour un joueur i est l'ensemble des distributions de probabilités sur l'action A_i décrite par l'opérateur simplex $\triangle$.

$$\triangle(\mathrm{A}_i) = \{ \mathrm{q}_i : \mathrm{A}_i \rightarrow [0,1] \mid \textstyle\sum_{i=0} q_i(a_i) = 1\}$$

pour des stratégies mixtes:

$$\mathrm{Q} = \textstyle\prod_i \triangle(A_i) \qquad et \qquad q = (q_i, q_{-i}) \epsilon Q$$

Le sens d'une stratégie mixte est que les actions sélectionnées peuvent réaliser un meilleur profit moyen, et que l'équilibre dans des stratégies mixtes est associé à la distribution de probabilités sur l'ensemble d'actions. Dans une stratégie mixte, les actions sont effectuées de manière aléatoire selon la fonction de distribution de probabilités.

Détermination du modèle de jeu

Comme déjà décrit, nous considérons dans notre modèle deux joueurs, le défenseur et l'attaquant. Nous supposons que l'attaquant et le défenseur ont des objectifs opposés. Les deux joueurs ont deux ensembles de stratégies différents: A décrit l'ensemble de stratégies de l'attaquant, et D décrit l'ensemble de stratégies du défenseur. Les joueurs reçoivent des gains pour toute action/stratégie choisie à partir de la fonction de gain. Nous considérons un jeu statique, où les décisions des joueurs pour choisir leur stratégie sont prises en même temps. Les stratégies choisies ne sont pas connues par l'opposant. Pour analyser le jeu il faut définir les objectifs des joueurs, les ensembles de stratégies et les fonctions de gains.

On peut définir les objectifs des joueurs ainsi: le défenseur s'efforce de maintenir l'efficacité de son système, tandis que l'attaquant cherche à diminuer l'efficacité du système et en même temps l'attaquant veut minimiser le risque d'être détecté par des mécanismes de défenses. Nous élaborons deux différents jeux: un jeu simple et un jeu avancé. Les ensembles de stratégies et les fonctions de gains sont définis par rapport au type de jeu.

Jeu simple:

Nous nommons ce type de jeu simple, parce que nous considérons les mécanismes de défenses simples (Détection des valeurs aberrantes) comme ensemble de stratégies du défenseur. L'ensemble de stratégies D du défenseur est comme suit:

> D : {Détection des valeurs aberrantes spatiales; détection des valeurs aberrantes temporelles; détection des valeurs aberrantes spatio-temporelles}

L'ensemble de stratégies A de l'attaquant est comme suit:

> A : {inflation 10% nœuds malveillants ; déflation 10% nœuds malveillants ; oscillation 10% nœuds malveillants ; inflation 20% nœuds malveillants ; déflation 20% nœuds malveillants ; oscillation 20% nœuds malveillants ; inflation 30% nœuds malveillants ; déflation 30% nœuds malveillants ; oscillation 30% nœuds malveillants}

Les fonctions de gains représentent les motivations et les objectifs des joueurs. Après la définition des objectifs des joueurs on constate que les deux joueurs veulent influencer la précision et l'efficacité du système. La précision du système peut être définie par les erreurs du système:

- L'erreur de prédiction $Error_{pred}$ mesure la précision globale du système en calculant la différence entre la latence mesurée et la latence estimée.
- L'erreur relative $Error_{rel}$ mesure la relation entre l'erreur de prédiction sous attaques et l'erreur de prédiction sans attaques.

Les fonctions de gains pour le défenseur sont comme suit:

- $P_{defenseur_{pred}} = -Erreur_{pred}$
- $P_{defenseur_{rel}} = \frac{1}{Erreur_{rel}}$

Pour définir les fonctions de gains de l'attaquant on veut aussi considérer qu'il doit investir un coût augmenté s'il compromet un grand taux des nœuds pour conduire l'attaque. En conséquence, on intègre le taux des nœuds malveillants dans les fonctions de gains de l'attaquant:

- $P_{attaquant_{pred}} = \frac{Erreur_{pred}}{\text{\% noeuds malveillants}}$
- $P_{attaquant_{rel}} = \frac{Erreur_{rel}}{\text{\% noeuds malveillants}}$

Jeu avancé:

Dans ce jeu, on définit des nouvelles stratégies pour le défenseur en utilisant un régulateur automatique. Ce régulateur permet d'avoir une sélection de seuil adaptive. La séléction du seuil adaptive est utilisée quand le système souffre sous une attaque forte pour réduire les conséquences de l'attaque sur la précision du système. Si l'erreur du système augmente, le seuil diminue, l'équation est présentée ci-dessous:

$$T_n = T_{n-1} - c\frac{(Erreur_{attaque}(n) - Erreur_{no\_attaque}(n))}{RTT_{Est}(n)} \tag{1.1}$$

Dans cette équation, c est une constante qui définit l'importance de l'erreur de prédiction. Il y a différentes possibilités pour sélectioner la valeur de l'erreur, soit on prend la moyenne ou des percentiles.

Les stratégies pour l'attaquant sont les mêmes qu'auparavant. L'ensemble de stratégies pour le défenseur est composé de différentes valeurs pour c : $\{0; 0.04; 0.06; 0.08; 0.1\}$ pour la boucle du régulateur et de différentes valeurs pour les percentiles de l'erreur de prédiction {25ieme; 50te; 75ieme} à prendre en compte.

Dans ce jeu, on a la fonction de gain pour le défenseur suivant:

- $P_{defenseur} = -Erreur_{pred}$

Pour l'attaquant, on a deux fonctions de gain différentes: la première parce qu'il veut diminuer la précision du système; et la deuxième parce qu'il ne veut pas diminuer le seuil car sinon le risque que l'attaque soit détectée augmente.

1. $P_{attaquant_{pred}} = Erreur_{pred}$
2. $P_{attaquant_T} = T_{moy}$

Résultats experimentaux

Pour analyser les différentes stratégies, on a exécuté des simulations dans le simulateur p2psim en utilisant les données fournies de King [45] et AMP [4]. On a exécuté toutes les combinations des stratégies de l'attaquant et du défenseur. Les résultats peuvent être consultés dans le chapitre 5.6.

Pour le jeu simple, on a trouvé que pour des réseaux larges (topologie King), l'attaque à l'inflation est l'attaque avec le plus grand impact sur le réseau. Pour le défenseur on a trouvé qu'utiliser la détection des valeurs aberrantes spatio-temporelles est le mécanisme de défense le plus efficace quand le seuil est au plus 1.5. Pour un seuil plus grand que 1.5, la détection des valeurs aberrantes spatio-temporelles et spatiales ont un impact similaire. En plus on a trouvé qu'utiliser un seuil de 1.25 résulte dans une plus grande efficacité qu'utiliser un plus grand seuil, en contradiction des travaux précédents [110, 111]. Pour les données d'AMP les résultats ne sont pas si homogènes, car pour la plupart des résultats une stratégie mixte est définie comme l'équilibre.

Pour le jeu avancé, on a montré que c'est mieux d'utiliser un seuil adaptif qu'un seuil fixe. La meilleure valeur pour le constant c est montrée d'être 0.8. Il faut noter qu'on n'a pas élaboré les attaques subtiles dans ce travail, comme c'est démontré que la détection des valeurs aberrantes n'est pas capable de détecter ces attaques.

1.3.2 Sécuriser les systèmes de coordonnées virtuelles à l'aide de l'apprentissage automatique

Dans cette section, nous montrons que nous sommes capables de détecter non seulement les attaques plus subtiles (l'attaque à l'inflation-lente et l'attaque à dégroupage du réseau) mais aussi les attaques simples (attaque à inflation, déflation, et oscillation). Avec notre méthode qui utilise l'apprentissage automatique nous développons un nouveau mécanisme de défense et nous éliminons alors les inconvénients des mécanismes précédents qui ne peuvent pas détecter les attaques subtiles. Pour utiliser l'apprentissage automatique, nous élaborons un ensemble de métriques, qui reflète le comportement du système.

Pour valider la méthode de détection nouvelle, nous analysons des nouvelles stratégies d'attaques plus complexes que les stratégies d'attaques déjà existantes. A l'aide d'une chaîne Markov, on crée des séquences d'attaques complexes pour lesquelles c'est plus difficile de reconnaître les structures d'attaques.

Notre méthode est validée par une détection déconnectée ainsi que par une détection en temps réel. Nous comparons trois différents algorithmes de l'apprentissage automatique supervisé d'une manière quantitative.

La structure de cette contribution est la suivante : premièrement nous définissons les stratégies d'attaques complexes. Ensuite, nous donnons une introduction à l'apprentissage automatique et nous définissons l'ensemble de métriques, avec lequel nous sommes capables de détecter toutes les stratégies d'attaques, surtout les attaques subtiles. Après cela, nous expliquons notre expérimentation et l'intégration dans le système réel.

Les stratégies d'attaques

Dans cette partie on montre tous les types d'attaques qui sont analysés dans notre expérimentation.

Les stratégies d'attaque simple:

Ici, nous considérons toutes les attaques simples (l'attaque à l'inflation, à déflation, et à l'oscillation) ainsi que les attaques plus subtiles (l'attaque à l'inflation-lente et à dégroupage de réseau).

Les stratégies d'attaque complexe:

- *Les attaques aléatoires:* Jusqu'à maintenant on a considéré des stratégies d'attaques où à partir d'un moment spécifique tous les nœuds malveillants conduisent le même type d'attaques. Pour les attaques aléatoires, les nœuds malveillants conduisent différents types d'attaques. Les nœuds malveillants choisissent l'attaque à exécuter aléatoirement.
- *Les scénarios à deux attaques:* Pour valider notre méthode de détection dans une première étape, nous créons des scénarios d'attaques où le temps des expérimentations est divisé en quatre tranches de longueurs pareilles. La première et la troisième tranche sont sans attaque. La deuxième tranche consiste dans la première attaque et la quatrième tranche consiste dans la deuxième attaque. Il y a trois scénarios à deux attaques qui en résultent: Déflation - Inflation-lente; Oscillation - Inflation; Dégroupage de réseau - Oscillation.
- *Les scénarios de séquences d'attaques:* A l'aide d'une chaîne Markov (voir figure 6.1), nous élaborons vingt différents scénarios de séquences d'attaques, mais pour simplifier l'analyse nous nous concentrons sur les scénarios les plus importants (le temps d'exécution consiste en 200 itérations):
  - *Séquence A*: pas d'attaques pendant 15 itérations; attaques à l'inflation pendant 15 itérations; attaques à dégroupage de réseau pendant 55 itérations; attaques à déflation pendant 35 itérations; attaques à l'inflation pendant 80 itérations. *Itérations totales sans attaques: 15.*
  - *Séquence B*: pas d'attaques pendant 10 itérations; attaques à l'inflation pendant 55 itérations; attaques à l'oscillation pendant 50 itérations; attaques à l'inflation-lente pendant 55 itérations; attaques à dégroupage de réseau pendant 30 itérations. *Itérations totales sans attaques: 10.*
  - *Séquence C*: pas d'attaques pendant 30 itérations; attaques à dégroupage de réseau pendant 35 itérations; attaques à l'inflation-lente pendant 35 itérations; pas d'attaques pendant 15 itérations; attaques à l'inflation-lente; attaques à l'inflation pendant 45 itérations. *Itérations totales sans attaques: 45.*
  - *Séquence D*: pas d'attaques pendant 40 itérations; attaques à l'inflation pendant 30 itérations; attaques à l'oscillation pendant 40 itérations; attaques à dégroupage de réseau pendant 40 itérations; attaques à l'inflation-lente pendant 35 itérations; pas d'attaques pendant 15 itérations. *Itérations totales sans attaques: 55.*
  - *Séquence E*: pas d'attaques pendant 50 itérations; attaques à l'inflation pendant 10 itérations; pas d'attaques pendant 50 itérations; attaques à l'oscillation pendant 65 itérations; attaques à l'inflation pendant 25 itérations. *Itérations totales sans attaques: 100.*
  - *Séquence F*: pas d'attaques pendant 55 itérations; attaques à dégroupage de réseau pendant 40 itérations; pas d'attaques pendant 15 itérations; attaques à l'inflation-lente pendant 45 itérations; attaques à l'inflation pendant 15 itérations; pas d'attaques pendant 30 itérations. *Itérations totales sans attaques: 100.*

L'apprentissage automatique et l'ensemble de métriques.

L'apprentissage automatique provenant de l'intelligence artificielle est utilisé pour apprendre des informations des données à une machine pour que cette machine puisse reconnaître des structures ou des règles concernant les données. Dans ce travail, nous utilisons l'apprentissage supervisé, car nous pouvons facilement obtenir des données nommées. Ces données nommées prédisent quel type de données une ligne spécifique fait partie. Dans notre cas, il y a deux classes, soit une attaque ou non-attaque. Nous analysons trois différentes méthodes de l'apprentissage supervisé : L'arbre de classification et régression (CART), l'arbre de décision C4.5, et les machines à vecteur de support (SVM). Les deux méthodes CART et C4.5 sont des arbres de décisions où des décisions sur les données sont prises par rapport à des règles créées soit analogues (CART) soit à base des entropies d'information. Les machines à vecteur de support sont utilisées pour représenter les données d'entrée dans un espace à d'autres dimensions. L'objectif est de rendre les données d'entrée séparables de manière linéaire. L'ensemble de métrique:

Le défi de ce travail est de définir l'ensemble de métrique, car cet ensemble doit être capable à détecter toutes les différentes attaques. Pour cette raison nous avons élaboré un ensemble qui prend en compte les relations temporelles entres les données consécutives. Nous avons trouvé qu'il existe une temporisation de quatre unités. Notre ensemble de métrique est alors comme suit:

1. *Feature A* est l'erreur local médian e_{median}.
2. *Feature B* est $\delta_1 = e_{median_t}$- $e_{median_{t-1}}$.
3. *Feature C* est $\delta_2 = e_{median_t}$- $e_{median_{t-2}}$.
4. *Feature D* est $\delta_3 = e_{median_t}$- $e_{median_{t-3}}$.
5. *Feature E* est $\delta_4 = e_{median_t}$- $e_{median_{t-4}}$.
6. *Feature F* est δ_{1_t}- $\delta_{1_{t-1}}$.
7. *Feature G* est $\mid \delta_{1_t}$- $\delta_{1_{t-1}} \mid$.

Expérimentations

Nous évaluons les différentes stratégies d'attaques pour l'algorithme de Vivaldi dans deux environnements différents. Premièrement, on considère le simulator p2psim en utilisant la topologie King, c.-à-d. nous avons 1740 nœuds. Deuxièmement, on considère le réseau PlanetLab avec 500 nœuds. Nous avons deux installations différentes pour notre méthode de détection : une installation globale et une installation locale. Pour l'installation globale, les informations de tous les nœuds sont enregistrées centralement et analysées ensemble. Par contre, pour l'installation locale, tous les nœuds décident d'une manière individuelle s'il y a une attaque ou non.

Pour l'installation globale on calcule l'ensemble des métriques à base de la valeur médiane de l'erreur locale des nœuds. Pourtant pour l'installation locale on prend l'erreur locale seulement des nœuds individuels. Nous utilisons les sources de weka pour en créer les classificateurs. Nous évaluons les résultats en calculant le pourcentage d'attaques qui sont classifiées correctement: les vrais positifs. Nous calculons aussi le pourcentage des itérations non-attaques qui sont classifiées comme attaque: faux positifs. Les résultats sont montrés dans le chapitre 6.5.

Pour les stratégies d'attaques seules et pour les scénarios à deux séquences dans l'installation globale on reçoit de très bons résultats pour les deux arbres de décisions, on a des vrais positifs 99% et des faux positifs d'environ 3%. On reçoit à peu près les mêmes résultats si on augmente le taux des nœuds malveillants. Pourtant, les machines à vecteur de support montrent de très grandes faux positifs (60%).

Pour les scénarios de séquences d'attaques les résultats divergent beaucoup et dépendent du taux d'itérations totales sans attaques. Si le taux est petit, surtout les faux positives sont larges. On déduit de ces résultats, qu'on a besoin au moins d'un taux de 25% d'itérations totales sans attaques pour qu'on puisse créer des arbres de décisions assez hétérogènes. S'il n'y a pas assez de lignes de données sans attaques le classificateur ne peut pas apprendre comment différencier ces lignes.

Pour être capable de comparer notre méthode avec les mécanismes de détecter des valeurs aberrantes nous examinons l'installation locale. On trouve que notre méthode fonctionne mieux que la détection des valeurs aberrantes.

Intégration dans le système réel

Pour les expérimentations démontrées auparavant, nous avons utilisé la classification d'une manière déconnectée, pourtant c'est important d'être capable de détecter les attaques en temps réel. Comme la classification est supervisée, nous entraînons notre algorithme d'une manière déconnectée. En intégrant l'arbre de décision dans le système réel nous pouvons détecter les attaques en temps réel. Les résultats pour l'intégration dans le système réel sont décrits dans le Chapitre 6.6.

Pour améliorer l'intégration locale, nous modifions la manière pour calculer l'ensemble métrique localement. Nous remarquons qu'utiliser seulement l'erreur locale du nœud lui-même ne donne pas des résultats satisfaisants, c'est pourquoi nous élaborons que tous les nœuds calculent l'ensemble de métriques à base des erreurs locales des nœuds avoisinés. Avec cette méthode, les résultats de la détection en temps réel sont beaucoup mieux. La performance de la détection en temps réel est aussi bien que la classification déconnectée.

1.4 Conclusion et travaux futurs

Dans cette thèse, nous avons deux contributions principales. La première contribution considère la théorie des jeux pour analyser systématiquement les stratégies d'attaques et de défenses. Nous avons étudié les interactions stratégiques des deux joueurs. Nous avons élaboré deux différents jeux, le premier pour analyser les stratégies de défenses simples et le deuxième pour vérifier les stratégies de défenses avancées. Pour les défenses avancées nous avons considéré un système à régulation automatique pour avoir une sélection adaptive du seuil.

La deuxième contribution est l'élaboration d'une nouvelle méthode qui détecte toutes les attaques contre les systèmes aux coordonnées virtuelles. De plus, nous avons créé des nouvelles stratégies d'attaques plus complexes. Dans ces stratégies d'attaques complexes, nous avons conceptualisé des scénarios à deux attaques ainsi que des scénarios de séquence d'attaques. Pour notre nouvelle méthode de détection nous avons utilisé des algorithmes d'apprentissage supervisé. Nous avons trouvé que les machines à vecteur de support, malgré leur réputation d'avoir une meilleure performance que les arbres de décision, réalisent tout de même une bien plus mauvaise performance pour les systèmes aux coordonnées virtuelles.

Comme travaux futurs, nous utiliserons différents modèles de jeux pour analyser les stratégies d'attaques et de défenses plus profondément. Jusque là nous avons utilisé un jeu statique, mais il existe des jeux plus complexes, comme les jeux répétitifs, où nous pourrions modéliser plusieurs interactions entre l'attaquant et le défenseur. Une autre approche serait d'appliquer un modèle d'apprentissage par renforcement. Les fonctions de gains seront alors plus complexes et nous pourrions évaluer un jeu plus réaliste.

En outre, nous avons utilisé des méthodes d'apprentissage supervisées, mais ces méthodes ont un grand inconvénient, elles requierent des données nommées, dont on est souvent difficilement garanti d'en avoir. Surtout dans les systèmes réels, il est difficile d'obtenir de telles données, comme on ne peut pas toujours savoir si une attaque est présente dans le jeu de données. Pour cette raison nous investiguerons des méthodes non-supervisées, cependant en sachant que ces méthodes souffrent souvent d'un manque de précision.

En plus d'améliorer nos méthodes existantes, nous adapterons nos méthodes pour d'autres systèmes qui sont similaires aux systèmes des coordonnées virtuelles. Les systèmes aux coordonnées virtuelles utilisent la latence pour créer des mesures et il existe d'autres systèmes qui utilisent aussi la latence pour des aspects différents. Le protocole pour synchroniser le temps sur les ordinateurs (NTP) utilise la latence pour choisir le meilleur serveur de référence. Nous investiguerons si les attaques contre les systèmes aux coordonnées virtuelles peuvent aussi affecter ce protocole et nous chercherons des méthodes pour prévenir le système de telles attaques.

Chapter 2

Introduction

It is not new that the Internet traffic is fastly growing. Following Visual Networking Index by Cisco for Forecast and Methodology [81], the global IP traffic increased eightfold during the last five years and they forecast that it will grow fourfold until 2015. A big amount of the Internet traffic is due to peer-to-peer (P2P) applications that have the goal to share distributed applications. P2P applications are used to create shared collaborative spaces or to share audio, video, or data content. A participating computer is called a node or a peer. Nodes in P2P applications provide resources for sharing and so the workload can be partitioned between the participating peers. When more nodes join a P2P network, the demand of resources is increasing but also the amount of available resources increases. Well known P2P applications exist in different fields, for instance BitTorrent [29] as a content distribution application, Skype [7] for voice-over-IP applications, or Chainsaw [78] for video broadcasting. The 2007 internet study of the IPoque team [52] showed that P2P traffic made almost 70% of the overall traffic in 2007 in Germany [89]. Although, the proportion of P2P traffic is decreasing in relation to the overall protocol distribution it still produces the most Internet traffic (Figure 2.1). Following [90] this decreasing phenomenon does not mean that there is less P2P traffic, it only means that P2P traffic is not growing as fast as other traffic.

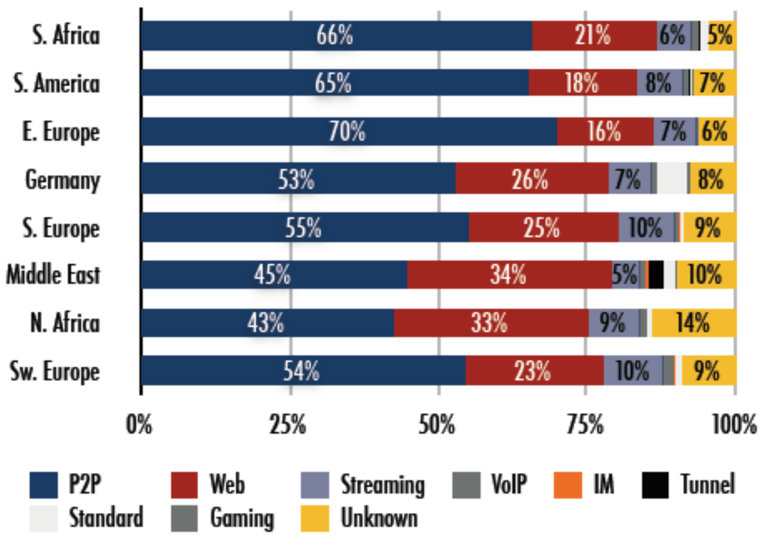


Figure 2.1: Protocol distribution out of the Internet Study 2008/2009 [90].

With the growth of networks, the performance becomes more and more important. In order to increase the efficiency of P2P networks, applications started to make use of location awareness so that nodes do not need to choose blindly from where to download from or with whom to communicate. Therefore, virtual coordinates systems have been developed as an overlay network that implements location awareness. In these systems, nodes obtain coordinates in a virtual space based on the relation of the latencies between them and neighbor-nodes. This way, a node's close neighbor-nodes have a low latency while nodes that are further away have higher latencies. The increased efficiency lies in the fact that a node will focus first on the physically close nodes to download data and with having a low latency, a node faster receives the network packets.

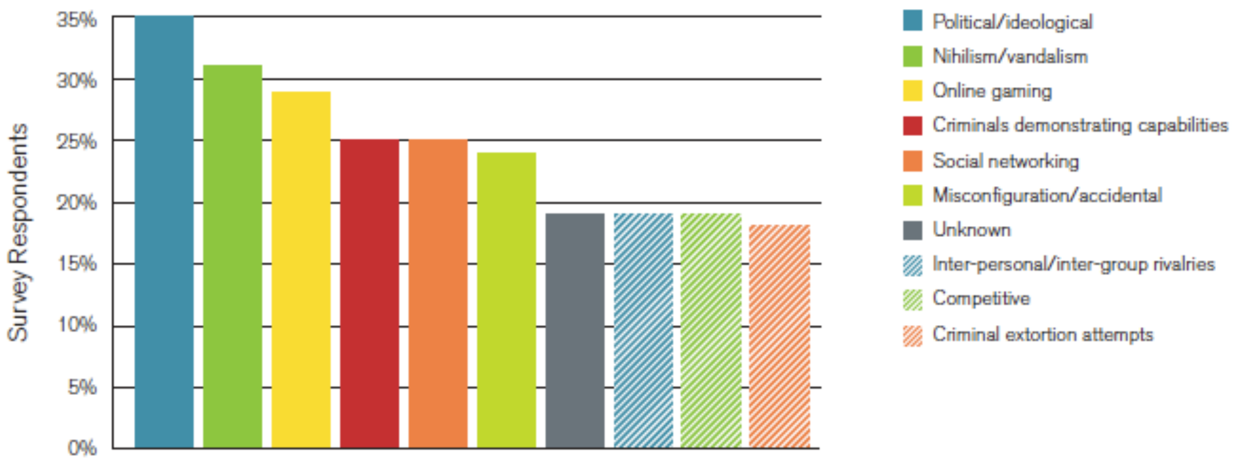


Figure 2.2: Attack motivations considered common or very common. Source: Arbor Networks, Security Report 2011 [73].

Unfortunately, security reports [72] show that attacks in the Internet become more frequent and that attackers increase their efforts. Furthermore, the motivation behind attacks is mostly the desire to vandalize after political or ideological attack motivations [73], as illustrated in Figure 2.2. Attackers with a motivation to vandalize those virtual coordinate systems, can degrade vastly the performance and accuracy of such systems. In virtual coordinate systems nodes have to report their location and the round-trip-times (RTT) towards neighbors in order to leverage a location awareness. Attackers can lie about their location and their RTT and consequently, the virtual coordinate system is not able to implement the correct and accurate location awareness. This can lead to a worse performance than using a P2P file sharing application without location awareness. Even though, this topic has been treated in research, open issues still remain and new attacks are elaborated. In this thesis, we address these issues and propose a new way for facing attacks in virtual coordinate systems.

2.1 Context and Motivation

The beginning on building virtual coordinate systems focussed on creating accurate predictions of latencies between nodes in order to implement a location awareness. The algorithms provided consider that nodes will cooperate and that the nodes are all benign with respect to reporting their latencies and coordinates. Nevertheless, soon after those different algorithms the first attacks were elaborated that could harm a correct implementation of the location awareness. Following, different mechanisms for detecting and mitigating the attacks have been proposed. Although, the validation of the defense mechanisms are always made through experiments. Therefore, we systematically study attack and defense techniques by developing a game theoretical model. This game model relies on well-known equilibrium concepts in order to assess the strategic interactions between the attacks and defenses. Game theory provides powerful tools that allow us to model an advanced adversary who knows how and what defense strategies are used and can adjust his attack strategies accordingly. This framework then allows us to draw out conclusions such as understanding what are the strengths and weaknesses of both defenses and attacks.

Furthermore, recent research [22, 23] identified new, more subtle and yet highly effective attacks that are able to bypass such existing defense mechanisms. During these attacks, malicious nodes lie about their coordinates only by small amounts and over time pushes benign nodes continuously move from their correct positions. Threshold-based defense mechanisms are vulnerable to these subtle attacks as they remain under the radar. No solutions to these attacks have been proposed to the best of our knowledge. We use supervised machine learning techniques to detect those subtle attacks and still being able to detect the the simple attacks as well. We provide a definition of an effective feature set that is able to identify even slow attacks as it considers a temporal correlation of the data set.

2.2 Contributions and Thesis Roadmap

This thesis is divided in three major parts, the structure and the contributions look as follows:

- In Part I, we give an overview of the state of the art related to our research area. So, we investigate the state of the art of virtual coordinate systems in Chapter 3. We investigate landmark-based systems that rely on infrastructure components for implementing a location awareness as well as completely decentralized virtual coordinate systems. After giving an overview of the different available algorithms, we explain in detail the decentralized algorithm Vivaldi [32] on which the contributions of this thesis are based on. Furthermore, we explore existing attacks and defense mechanisms against virtual coordinate systems, specifically against Vivaldi in Chapter 4. We point out the weaknesses of the existing defense mechanisms and explain why subtle attacks cannot be detected by those mechanisms.

- The contributions are described in Part II. We define the game theoretical framework for defining the optimal defense mechanisms and attack strategies in Chapter 5. We consider a Byzantine adversary that controls a percentage of the nodes in the system and conducts three different types of attacks: inflation, deflation, and oscillation. These three attacks correspond to when a node reports large coordinates far away from the origin, small coordinates near the origin, and randomly chosen coordinates, respectively. We consider defense strategies based on outlier detection since they have been shown to provide good results under the assumption the attacker does not know the defense strategy [111]. Specifically, we assume the defender attempts to mitigate the attacks using three defense techniques based on outlier detection: spatial, temporal, and spatial-temporal. The defense techniques also use realistic system design assumptions that make them easily integrated into current virtual coordinate systems, *i.e.*, they do not rely on the triangle equality [30], do not require extra node sets and network communication [94, 95], and do not require trusted parties [54, 88].

  A critical component of an outlier detection mechanism is the threshold that is used to decide if a data point is accepted by the system or is suspected of coming from a malicious node. Many outlier detection schemes use a fixed threshold, usually determined experimentally. Such an approach is inflexible, prone to errors, and may be exploited by an adversary to remain undetected. We leverage control theory to design an adaptive threshold technique to improve the threshold selection and include outlier detection mechanisms based on adaptive thresholds in our study. Our contributions include:

– We model rational attackers in virtual coordinate systems using the Nash equilibrium and irrational attackers using the quantal response equilibrium. From the defender side, we use game theory to tune our defensive mechanisms in order to mitigate the attacks.
– Using our framework, we determined that for large networks (*i.e.*, the King topology), the inflation attack has the greatest impact on the system. To defend the system, we find that spatial-temporal outlier detection is the most effective technique given lower spatial outlier thresholds (*e.g.*, ≤ 1.5) and both spatial-temporal and spatial outlier detection provide similar defense performance for higher thresholds. Furthermore, our analysis finds that, independent of the game strategy or the error metric selected, a spatial outlier threshold of 1.25 results in the best system performance, which is smaller than the value found in previous work.
– We found that the resulting strategy profiles for smaller networks (*i.e.*, the AMP topology) are not as homogeneous as those for the larger King topology, with most of the resulting strategy profiles consisting of a mixed strategy. For example, given the spatial outlier threshold of 1.75, the attacker has the greatest payoff while applying all three attacks with their given probabilities using only 10% malicious nodes. The countermeasure profile looks similar, applying each of the three defense techniques. Both the percentage of malicious nodes necessary to efficiently create the greatest negative impact and the attack and defense profiles have not previously been systematically explored.
– We found that when comparing strategies using a fixed threshold with strategies using an adaptive threshold selection for the outlier detection, the adaptive threshold is more effective in defending against attacks than a fixed threshold. Our analysis shows that when an attacker has as goal disturbing the network as much as possible, using inflation with 30% attackers is the best attack strategy. If the attacker wants to remain also undetected then oscillation and deflation attacks with 10% attackers are the best rational choice. We found that the best parameters for the adaptive threshold is to use the 75^{th} percentile of the prediction error and with a value for the constant c of 0.08 to update the threshold, where c is a system parameter that captures the importance given to the prediction error when updating the threshold.

Furthermore, we define a new detection method using supervised classification in order to detect the more subtle frog-boiling and network-partition attacks in Chapter 6. We consider the detection of all existing attacks against decentralized virtual coordinate systems by leveraging supervised machine learning methods: decision trees and support vector machines. Our approach is able to detect and mitigate all known attacks used in both single attack strategies where individual attacks (frog-boiling, network-partition, oscillation, inflation and deflation) are launched by an attacker and more complex attack strategies, where successive attack phases are intermixed without assuming any fixed order in the attack sequence. Our contributions are as follows:

– We propose a practical method to counter the frog-boiling and network-partition attacks, or any complex attack strategy in which several individual attacks are launched by a powerful adversary. For example, the latter can combine several single attacks following a Markov chain model.

  - We develop a feature set, based on a node's local information, for embedding it into a multidimensional manifold in order to reveal attacks. This process has resulted in seven feature variables that prove to be the most relevant for the prediction and classification task.
  - We provide a quantitative analysis of supervised machine learning methods, *i.e.,* decision trees and support vector machines, for detecting all known attacks in an offline analysis scenario. We evaluate our techniques using the Vivaldi [32] virtual coordinate system through simulations using the King data set and real deployments on PlanetLab. Among the two different machine learning techniques, decision trees and support vector machines, decision trees are able to mitigate all known attacks, outperforming support vector machines by achieving a much lower false positive rate. Our approach works both in a global manner, where all nodes actively exchange local information and a collective decision is taken, as well as in an individual manner, where each node locally decides whether an attack is occurring or not. The results for simulations using the King data set and for real deployments on PlanetLab both demonstrate that our approach identifies the different attacks with a $\sim 95\%$ true positive rate.
  - We validate our method by analyzing the performance of online detection. We integrate decision trees into the Vivaldi VCS in order to detect attacks in real-time. We analyze the effectiveness of the real-time detection in both a global manner, where a collective decision is taken, as well as in a local manner, where each node decides by itself if an attack is occuring or not. In order to improve the local real-time detection, we design a new way for a node to decide if an attack is occurring. This is accomplished by taking into account not only a node's local information but also the information already being collected through its interaction with those in its neighbor set. The real-time detection validates the performance we achieved during offline classification with a $\sim 95\%$ true positive rate for the global manner, and a $\sim 90\%$ true positive rate for the improved local manner.

- Finally, we conclude this work in Part III, where conclusions and future perspectives are discussed in Chapter 7. We discuss limitations of our approaches and how to overcome these. We explore how to enlarge the proposed methods and how these methods could be deployed in a different context or protocol.

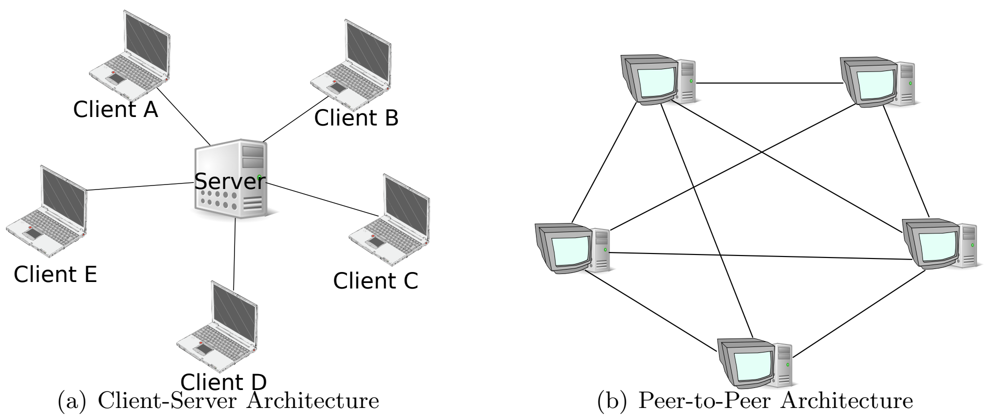
Part I

State of the Art

Chapter 3

Virtual Coordinate Systems

Scalability is an important aspect when designing networks, as todays networks need to be operable with a large amount of components or users. Therefore, the traditional client-server (see Figure 3.1(a)) architecture becomes less applicable and more often replaced by peer-to-peer architectures. Peer-to-peer (P2P) is a distributed application architecture where many nodes are interconnected as shown in Figure 3.1(b) and all the nodes (a.k.a peers) act as server and client simultaneously in order to share the workload among themselfs. All the peers have the same responsibilities, there are mostly no hierarchical 'higher' peers. This architecture avoids the single point of failure (like the server in the client-server architecture), as files are transferred directly between the peers.



(a) Client-Server Architecture (b) Peer-to-Peer Architecture

Figure 3.1: Network Architectures

P2P architectures have become famous due to file sharing services (Kazaa, BitTorrent, etc.). Of course, many different sharing architectures have been proposed with the goal to distribute and locate desired files. The most common approach is the usage of distributed hash tables (DHT) [84, 98]. DHTs provide a structured routing algorithm for the file lookup. DHTs store key-value pairs in a distributed manner. Chord [98] is a distributed lookup protocol that maps a key onto a node. A key can then be related with a file, the key and that file are stored on the mapped node. Each node maintains a routing table (a.k.a finger table) in order to locate the node that is mapped with a given key.

It is common, that files are stored on many replication servers in order to ensure availability of

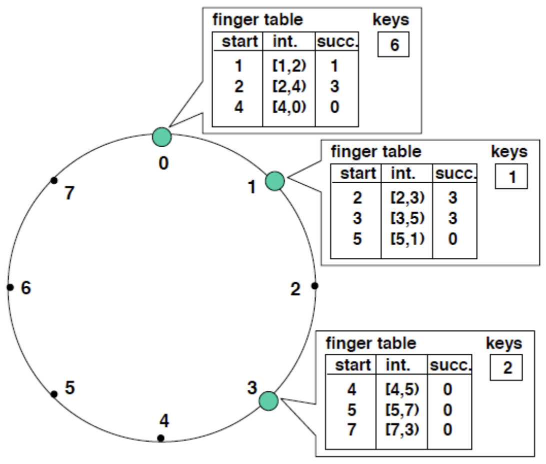


Figure 3.2: Chord: Finger tables and key locations for a network with nodes: 0,1,3 and keys: 1,2,6 [98]

the files. As there are many replicas available a peer has to choose from which replication server to download. The efficiency of those file-sharing systems can be improved by connecting to a node to whom the latency is shorter and the bandwidth is larger rather than connecting to a node with higher latency and smaller bandwidth. In other words, one can use these features of network information to take advantage of network locality, where based on latencies or available bandwidth an overlay network topology is created. However, active measuring the network latencies generates high costs and the related overhead can outmatch the achieved gain.

Therefore, virtual coordinate systems have been proposed with the key idea to represent the network in a geometric space. Every node is associated with a position in that space and the distances to its neighbors are in relation with the latency between the nodes. The major contribution of virtual coordinate systems is that a node does not need to undertake additional measurements for calculating the distances to other nodes. Network overhead is avoided by sending information of the node position within the normal messages. The virtual coordinate system builds a location-aware overlay network.

Overall, there are two different families for the location-aware overlay networks. There are overlay networks that need to use some kind of landmark, infrastructure components, to be able to provide location-awareness, these are called landmark-based systems. The other type of coordinate systems usually work in a complete decentralized way and do not have the need for any landmark infrastructure for its service. These are called decentralized Virtual Coordinate Systems.

The virtual coordinate systems could be based, as already mentioned, on two different network-variables, either the latencies or available bandwidth. However, as those overlay networks want to avoid additional message overhead for constructing the coordinate system, all the coordinate systems presented in this work use the latency as factor for the coordinate system creation. The latency, which is often considered as the round-trip-time (RTT) between nodes, is contained in normal network messages (e.g. ICMP ping messages).

In Section 3.1, we give an overview of the landmark-based Systems. In Section 3.2, we explore the decentralized Virtual Coordinate Systems and we explain Vivaldi, the decentralized virtual coordinate system in more detail in Section 3.3 as we focus our work on this system.

3.1 Landmark-based Systems

Landmark-based systems rely on infrastructure components (such as a set of landmark servers) to predict distances between any two hosts. In [75], Ng and Zhang propose a Global Network Positioning (GNP) mechanism. In this network, the nodes obtain absolute coordinates. This is done in two steps, the first step is that a predetermined set of landmark-nodes compute their own coordinates in a chosen geometric space. In a D-dimensional space, at least D+1 landmarks are needed. The landmarks calculate their inter-landmark Round-Trip-Times (RTT) using ping messages and take for each path the minimum of several measurements. During the second step, ordinary hosts calculate their own coordinates based on the coordinates of the landmarks and the RTTs to the landmarks. Again, the minimum RTT of several measurements are considered. Similar to this method, IDES [68] calculates the coordinates of the landmarks, but here the landmarks report the distances to a central server. The authors of [75] provide an accuracy evaluation that shows, that GNP has a very low relative error. The good accuracy is confirmed by other approaches that compare their accuracy to this one.

In NPS [76], Ng and Zhang extend the GNP framework by adding a third type of nodes in their system; the membership servers. This addition to the original algorithm is done in order to avoid that landmark-nodes become bottlenecks. The membership servers store system configuration parameters and maintain information about nodes. Furthermore, the membership servers are used to define reference nodes out of ordinary hosts. Those reference nodes can then be used, besides the existing landmarks, for new nodes to base their coordinates-computation on them. The membership servers select randomly among all available nodes that have finished their position calculation for reference nodes.

Tang *et al.* [99] propose a faster embedding of network distances for these architectures by employing the Lipschitz embedding [18]. This goes with the assumption that two close nodes have similar distances to other nodes. In contrast to using absolute coordinates, IDMaps [40] uses Tracers to calculated the original distance between them and Address Prefixes in order to establish a coordinate system.

All those proposed architectures have the need of pre-defined landmark nodes, however in Lighthouses [80], the authors propose a new way of using landmarks without the need for these nodes to be predefined, but randomly selected. Furthermore, the bottleneck limitations of using landmarks is addressed by using several landmark sets instead of one common landmark set in [80] and [64].

In order to minimize the differences between the measured RTTs and the estimated RTTs, [75, 76, 80] apply the Simplex Downhill method [71], whereas, [68] use Singular Value Decomposition and Non-Negative Matrix Factorization [60]. The Singular Value Decomposition is also used by [64]. [64] and [99] apply both Principle Component Analysis (PCA) [53] for measuring distances.

Treeple [23], while not strictly coordinate based, provides secure latency estimation, using landmarks as vantage points for providing traceroutes on the Internet. In Treeple, landmarks perform traceroute measurements to peers, which the landmarks can then digitally sign and provide for nodes to compute the network distance themselves. In order to be secure, the authors of Treeple make the strong assumption that no routers can be affected by an attacker only end-hosts can.

All those proposed methods rely on infrastructure components to act as landmark-nodes. These landmarks represent a single-point of failure issue. Even if this issue is faced in a few approaches [64, 76, 80], the usage of landmarks limits the scalability of those approaches. To overcome these issues, decentralized methods have been proposed and are elaborated in the following Section.

3.2 Decentralized Virtual Coordinate Systems

Decentralized virtual coordinate systems, i.e. PIC [30], Vivaldi [32], and PCoord [62, 63], are designed to efficiently create and maintain a stable set of coordinates that accurately predict the latency between nodes without the need for fixed infrastructure nodes, and without any additional message overhead.

All three approaches use some kind of reference set, on which a node calculates the distances to. The idea of the decentralized virtual coordinate systems is that there is no need to learn the distances between all arbitrary nodes, but only between a few selected nodes, the reference set. A node has to select the reference set prior or after joining the system. This selection can happen in different ways. A node can just randomly select other nodes or can consider topology information for this process. PIC [30] shows that choosing half of the reference set to be physical close nodes and the remaining half random nodes, is the most efficient approach for selecting the reference set. Following this, Vivaldi [32] chooses the reference set the same way. In PIC this reference set is used for bootstrapping, whereas in Vivaldi all the nodes start at the origin.

PCoord [62] defines three different ways of joining the coordinate system. The first and second approaches handle the way reference sets are chosen prior the node joins the system. The first approach is to solely select random nodes. The second approach takes topology information and divides the nodes in clusters relative to the topology information, and a node has to choose among the clusters randomly. In the third approach to join the system, a node undergoes an iterative process without the need of a reference set beforehand, but joins at the origin. After the node joined the system, it will choose a reference set based on triangle distances at each iteration to compute distances and refine its own coordinates. The three approaches do also differ considering the amount of nodes in the reference set; PIC uses 32 nodes, Vivaldi uses 64 nodes, and PCoord refers to 10 nodes.

Besides the reference set, another characteristic is important and differently chosen for the three approaches, namely the latency prediction method. Vivaldi, PIC and PCoord use Euclidean spaces, but also Hyperbolic spaces have been treated [65, 93] but have been shown not to outperform the Euclidean spaces [65].

Once a node has joined the system, it keeps updating its coordinates in order to adapt to network changes and to minimize the difference between the actual measured RTT and the estimated RTT. This minimization is handled divergently, so PIC and PCoord use the Simplex Downhill method [71]. Vivaldi minimizes the squared error of prediction. All three approaches [30, 32, 62] evaluate the accuracy of their approaches by comparing the outcome with the accuracy of GNP [75] and state that their accuracies are as good as the accuracy of GNP. Donnet et al [36] state that Vivaldi is probably the most successful coordinate system, as it does not require any fixed network infrastructures and it does not differentiate between the hosts. In Section 3.3, we explore the algorithm of Vivaldi in detail.

Accuracy is being improved in Pharos [27], where a node will not only receive one set of coordinates, but a few according to how many distance ranges one wants to address. One set of coordinates positions the node at a global scale, while the others position the node at a smaller distance scale. One can see this approach as an improvement to Vivaldi, as they use the Vivaldi algorithm for all the clusters. Furthermore, accuracy is also addressed in Phoenix [26]. The authors propose an enhanced and decentralized version of landmark-based IDES [68] by removing some of its flaws.

3.3 Vivaldi

We selected the representative virtual coordinate system, Vivaldi, since it is a mature system, conceptually easy to understand and visualize, and has been shown to produce low error embeddings and offers a good tradeoff between performance and overhead [32, 65].

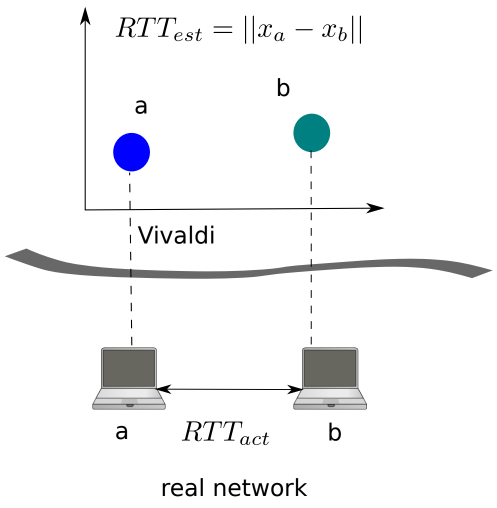


Figure 3.3: Vivaldi assigns each node synthetic coordinates

Vivaldi assigns each node synthetic coordinates (illustrated in Figure 3.3) in a multi-dimensional Euclidean coordinate space. The coordinates are assigned so that the distance between two nodes predicts the round-trip-time (RTT) between the two hosts. In order to accurately predict the RTT, Vivaldi proposes to minimize the error of prediction $Error_{pred}$.

$$Error_{pred} = |RTT_{Act} - RTT_{Est}|$$

where RTT_{Act} is the measured RTT between two nodes and RTT_{Est} is the RTT computed using the coordinates derived by the virtual coordinate system. Intuitively, the lower the system prediction error, the more accurate are the predicted RTTs.

Vivaldi describes a spring relaxation problem in which each pair of neighbor nodes is attached by a spring and the current length of the spring is the estimated RTT between the nodes. Tension on the logical springs causes the nodes to move through the coordinate space as each node attempts to minimize the difference between current spring lengths (RTT_{Est}) and the spring lengths at rest (RTT_{Act}). By minimizing the tension across all of the springs in the network, the protocol minimizes the error for the system. Thus as with real springs, if a spring is compressed it applies a force that pushes the nodes apart and if the spring is extended the spring pulls them together. Over time, the tension across all springs is minimized, and the position of each node produces the resulting coordinate.

Specifically, the Vivaldi protocol works as follows. Initially, each node i is assigned a random coordinate and establishes a reference set of peer nodes with which to exchange periodic updates. As nodes communicate with their reference set peers, they receive latency information that is used to update their coordinates. Algorithm 1 shows how a node i updates its coordinate x_i and error e_i as a result of minimizing the tension of the spring with remote node j. Node i updates its own coordinate

```
Algorithm 1: Vivaldi Coordinate Update
  Input: Remote node observation tuple ($(\langle x_j, e_j, RTT_{ij}\rangle)$)
  Result: Updated local node coordinate and error ($(x_i, e_i)$)
1 $w = e_i/(e_i + e_j)$
2 $e_s = |\|x_i - x_j\| - RTT_{ij}|/RTT_{ij}$
3 $\alpha = c_e \times w$
4 $e_i = (\alpha \times e_s) + ((1 - \alpha) \times e_i)$
5 $\delta = c_c \times w$
6 $x_i = x_i + \delta \times (RTT_{ij} - \|x_i - x_j\|) \times u(x_i - x_j)$
```

and error based on the tuple consisting of the remote node's coordinate x_j, the remote node's relative error with respect to its coordinate, e_j (both directly reported by node j), and the latency from node i to node j, RTT_{ij} (measured by node i). First, the algorithm calculates the observation confidence w (line 1) and relative error e_s (line 2). The relative error e_s expresses the accuracy of the coordinate in comparison to the measured network latency. Next, node i updates its local error (line 4) using an exponentially-weighted moving average with the weight α (line 3). Finally, the node calculates the movement dampening factor (line 5) and updates its coordinate (line 6). Both c_e and c_c are constants acting as system parameters.

As the nodes update their coordinates and the system stabilizes, the average system error is on the order of a few percent. Once the coordinate system has stabilized, the latency (*i.e.*, RTT) between two nodes is trivially estimated by computing the Euclidean distance between their coordinates.

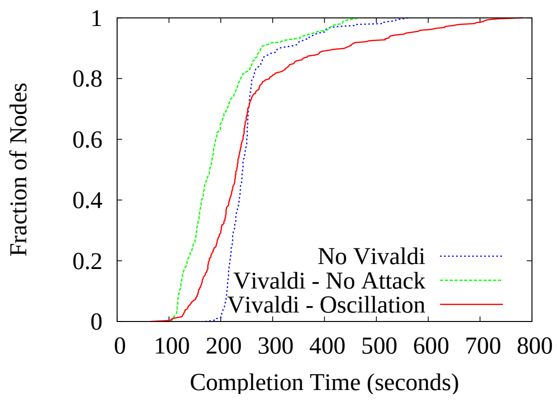


Figure 3.4: BitTorrent improves performance when configured to use Vivaldi, 315 nodes; attacks disrupt the performance (10% malicious) - PlanetLab

To show how the performance can be improved, we evaluated the file-sharing BitTorrent system [29] in a real-life PlanetLab [28] deployment of 315 nodes, We compare three scenarios in Fig. 3.4: **No Vivaldi**, the scenario where the BitTorrent tracker does not use virtual coordinates, but simply chooses nodes at random; **Vivaldi - No Attack**, the scenario where the tracker is coordinate-aware, i.e. when a client requests other peers to download from, the tracker will respond with a selection of nodes that are near the coordinate of the requesting node; **Vivaldi - Oscillation**, the scenario where the coordinates used by BitTorrent are impacted by an oscillation attack against Vivaldi, attacks against Vivaldi are explained in detail in Section 4.1. In our implementation, malicious nodes report randomly chosen coordinates and increase the RTT by delaying probes for up to 1 second. As can be seen in Fig. 3.4, when the tracker is aware of coordinates, the download times decreases by 50% for some nodes. However, when under attack, much of the gains brought on by the coordinates are lost, and for over 25% of nodes, the download times actually increase over the scenario when no virtual coordinates are used to optimize peer selection.

3.4 VCS Real Deployment

Virtual Coordinate Systems have been considered for real deployement, so Pyxida [82] offers an open source library and application that applies the virtual coordinates as proposed in Vivaldi [32]. One can either use Pyxida as a standalone version or on can use the Pyxida library. It provides following functionalities:

- Latency estimation between hosts.
- Closest rout to a host.
- Standalone application constructing a virtual coordinate system according to Vivaldi.

Pyxida is implemented within the Azureus (now called Vuze) [5] BitTorrent Client and enables to efficiently choose among potential data-providers. It is also used by PlanetLab [28], the global research network, in order to keep track of the coordinates of its machines.

In [59], the creators of Pyxida analyze the Pyxida integration in Azureus in large-scale and introduce a new way for managing the neighbor set in order to improve accuracy of the real deployed system. Furthermore, [97] provide an evaluation of the Vivaldi system in the real deployement in Azureus. Steiner *et al.* show that Azureus does well for predicting the RTT between hosts.

3.5 Overview VCS

This Chapter showed how Virtual Coordinate Systems can improve P2P architectures in order to be more effective in choosing neighbor peers in terms of latency between the peers. This chapter showed the different approaches how this coordinate systems can be elaborated either by having a landmark-based system or a decentralized system. An overview of the different approaches elaborated over the years is shown in Table 3.1.

Table 3.1: VCS - Overview

	Landmark-based	Decentralized	Predefined node-set needed	Year	Method
IDMaps [40]	X	-	-	2001	k-HST/k-center/t-spanner
GNP [75]	X	-	X	2002	Simplex Downhill
NPS (GNPv2) [76]	X	-	X	2004	Simplex Downhill
Tang et al. [99]	X	-	X	2003	PCA
Lim et al. [64]	X	-	-	2003	PCA/ Singular Value Decomposition
Lighthouses [80]	X	-	-	2003	Simplex Downhill
PIC [30]	-	X	-	2004	Simplex Downhill
Vivaldi [32]	-	X	-	2004	Squared error process
PCoord [62, 63]	-	X	-	2004/2006	Simplex Downhill
IDES [68]	X	-	X	2006	Singular Value Decomposition/ Non-Negative Matrix Factorization
Pharos [27]	-	X	-	2007	Squared error process
Phoenix [26]	-	X	-	2009	weighted non-negative least squares module
Treeple [23]	X	-	-	2010	Traceroute

Chapter 4

Attacks and Defenses on Virtual Coordinate Systems

In virtual coordinate systems, the well functioning is based on the assumption that the nodes are honest and do report correct coordinates and latencies. The well functioning of a system is described by good accuracy and stability. However, dishonest hosts can degrade those features with different goals. A malicious node can, by lying about its coordinates and latencies, isolate a group of nodes that rely on the updates of that malicious node. Furthermore, malicious nodes also can lead to instability in the system, so that the latency estimations are completely disordered and result in important performance decrease.

Landmark based systems can be protected against such malicious behavior, by securing the set of landmark-nodes. However, this is not feasible in a decentralized system as any node can act as a reference node and one can not guarantee that no infiltration of malicious users can happen.

We focus in this work on the virtual coordinate system Vivaldi as it is the most widespread system and because it produces low errors. Therefore, we do not reference attacks or defenses specifically related to other virtual coordinate systems. However, one can consider that attacks and defenses for Vivaldi might also be applicable on other virtual coordinate systems. There exists also work on securing other virtual coordinate systems, for instance PIC [30] provides a basic approach to avoid lying nodes based on the assumption that a node who is forging its coordinates, probably violates the triangle inequality.

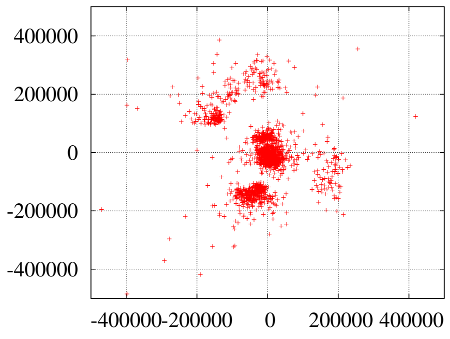
The remainder of this chapter looks as follows: In Section 4.1, we describe the existing attacks against the decentralized virtual coordinate system Vivaldi. We explore existing defense mechanisms in Section 4.2 against those attacks.

4.1 Attacks

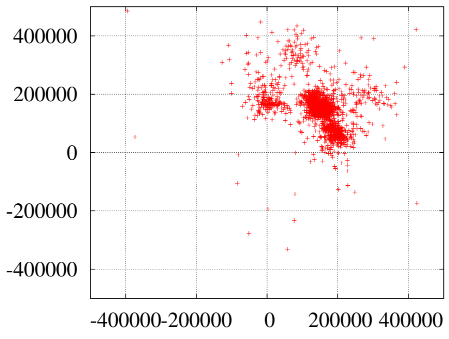
The attacks proposed in [22, 55, 56, 111] consider network intruders that can perform insider attacks and falsify the coordinates and latency reports. Kaafar *et al.* [55, 56] identify attacks, where an attacker can isolate a node, by reporting to him high coordinates with a low error. Furthermore, Zage *et al.* [110, 111] identify more concrete attacks in relation with how the attacker is lying about its coordinates and latencies, namely the inflation attack, deflation attack and oscillation attack, explained in the following paragraphs. Besides these three attacks, Chan-Tin *et al.* [22] propose two more subtle attacks, the frog-boiling attack and the network-partition attack. These two attacks are deviations of the inflation attack and are also explained in the following paragraphs. For these insider attacks, we consider that a malicious user can infiltrate several participating nodes in the system and manipulate their behavior.

Inflation Attack

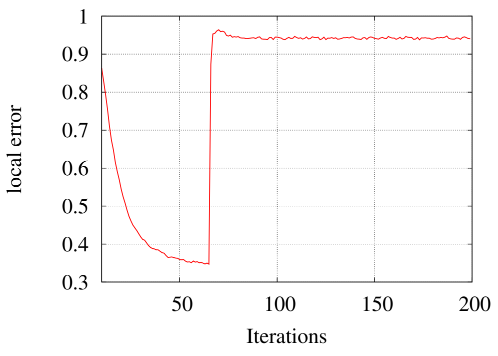
Within the coordinate inflation attack [110, 111], an attacker can push away a victim node from its correct coordinates. To achieve this, an attacker can pull it closer or push towards a location defined by the attacker. To pull the victim node towards a location, an attacker reports a low error and a small RTT, smaller than the estimated RTT. To push a victim node towards a location, the attacker reports coordinates close to the victim's coordinates, but with a high RTT, so that the victim node moves away from the reported coordinates. The larger the RTT, the larger the victim node will move away from its correct coordinates. We can see the impact of the inflation attack on the coordinates of the nodes in the system in Figure 4.1(b) when compared to the system in a benign environment 4.1(a), the nodes are pushed far from their correct positions. We can see the impact on the local error in Figure 4.1(c), where the error increases drastically when the attack starts at iteration 60.



(a) Coordinates: No Attack



(b) Coordinates: Inflation Attack

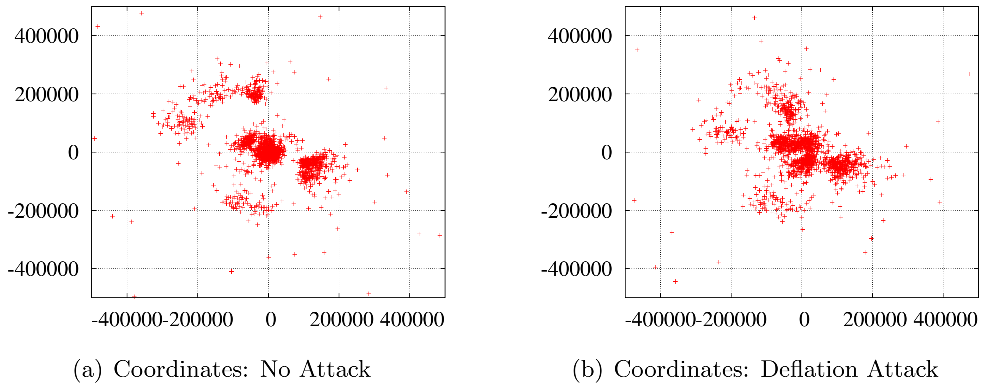


(c) Local Error

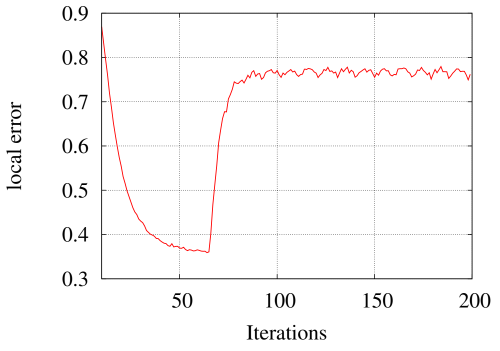
Figure 4.1: Impact of Inflation Attack

Deflation Attack

A coordinate deflation attack [110, 111] has as objective to render a victim node immobile. The victim node is not able to perform coordinate updates in order to find its correct position. To perform such an attack, a malicious node needs to report either coordinates so that the estimated RTT ($||x_i - x_j||$) matches the measured RTT and the difference will be zero, so the node will remain with its old coordinates. Or an attacker can influence the reported RTT so that it matches the estimated RTT. In Figure 4.2 the impact of a deflation attack, Figure 4.2(a) shows the coordinates of the nodes under normal circumstances, whereas Figure 4.2(b) shows the coordinates during the deflation attack, and we see that the nodes move towards the origin. The impact on the median local error is shown in Figure 4.2(c).



(a) Coordinates: No Attack (b) Coordinates: Deflation Attack

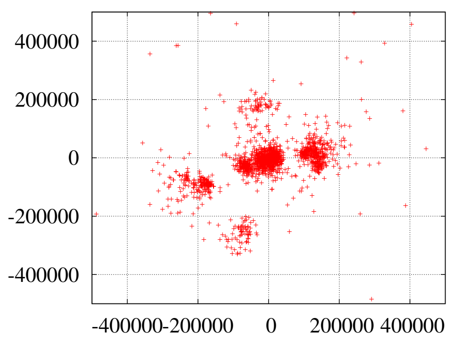


(c) Local Error

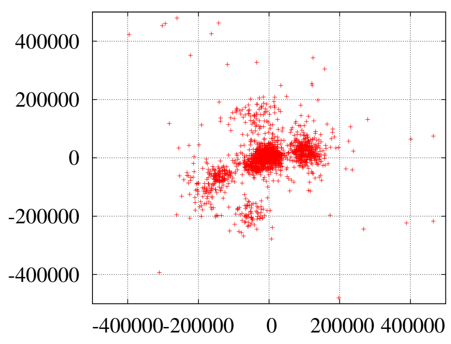
Figure 4.2: Impact of Deflation Attack

Oscillation Attack

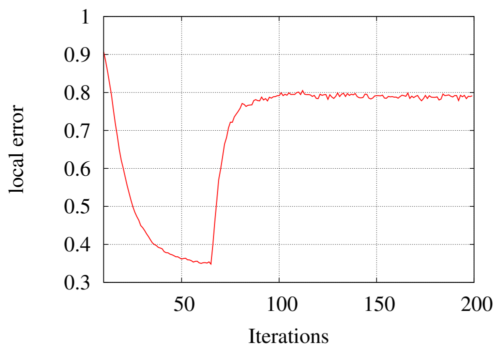
A coordinate oscillation attack [110, 111] results in a complete chaos in the network, as the nodes keep changing their coordinates and they do not converge to their correct positions. The impact of this attack is shown in Figure 4.3. The change in the coordinate-space to chaos is illustrated in Figures 4.3(a) and 4.3(b). This attack is performed by malicious nodes that report randomly wrong coordinates, random local error values, and random RTT. This leads to a maximization of the system error. The impact on the error is shown in Figure 4.3(c).



(a) Coordinates: No Attack



(b) Coordinates: Oscillation Attack



(c) Local Error

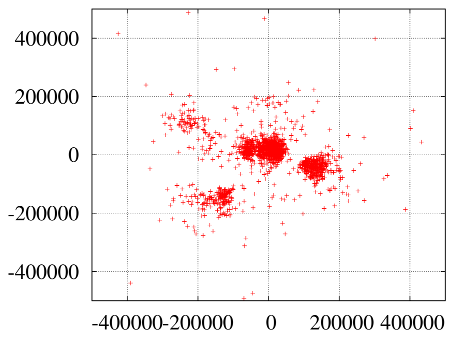
Figure 4.3: Impact of Oscillation Attack

Frog-Boiling Attack

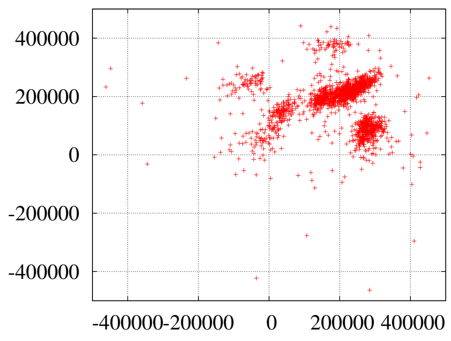
The frog-boiling attack [22] describes a derivation of the inflation attack, but compared to the inflation attack, this attack uses very small steps in order to remain under the radar of defense mechanisms that use any kind of thresholds. One can think of this attack as a slow-inflation attack. It is called frog-boiling attack as this term is often used as a metaphor to describe that somebody is not aware of slow changes. Figure 4.4(c) shows the impact of frog-boiling impact on the median local error of all nodes in the system. The attack starts around iteration 60, we see that the error value does increase but much slower than for the inflation attack (Figure 4.1(c)). Furthermore, Figure 4.4(b) shows that the frog-boiling attack has the same impact on the coordinates of the nodes than the inflation attack in comparison to the coordinates without an attack in Figure 4.4(a).

Network-Partition Attack

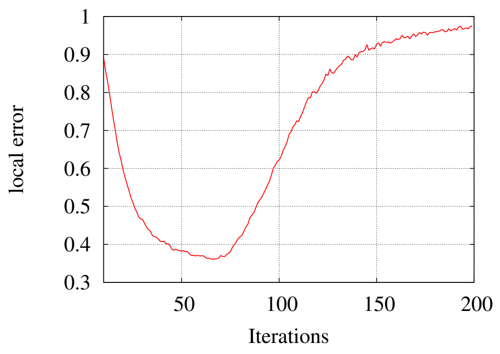
The network-partition attack [22] is an extension to the frog-boiling attack. An attacker performs this attack in order to divide a network, by having a group of malicious nodes pushing a subset of nodes in the network in one direction and another subset of nodes in the opposite direction. Figure 4.5(c) shows the impact on the median local error value, we see that it is similar to the frog-boiling impact. However, one can see that in Figure 4.5(b) two groups of nodes is built in comparison to the normal coordinate distribution in Figure 4.5(a).



(a) Coordinates: No Attack



(b) Coordinates: Frog-Boiling Attack



(c) Local Error

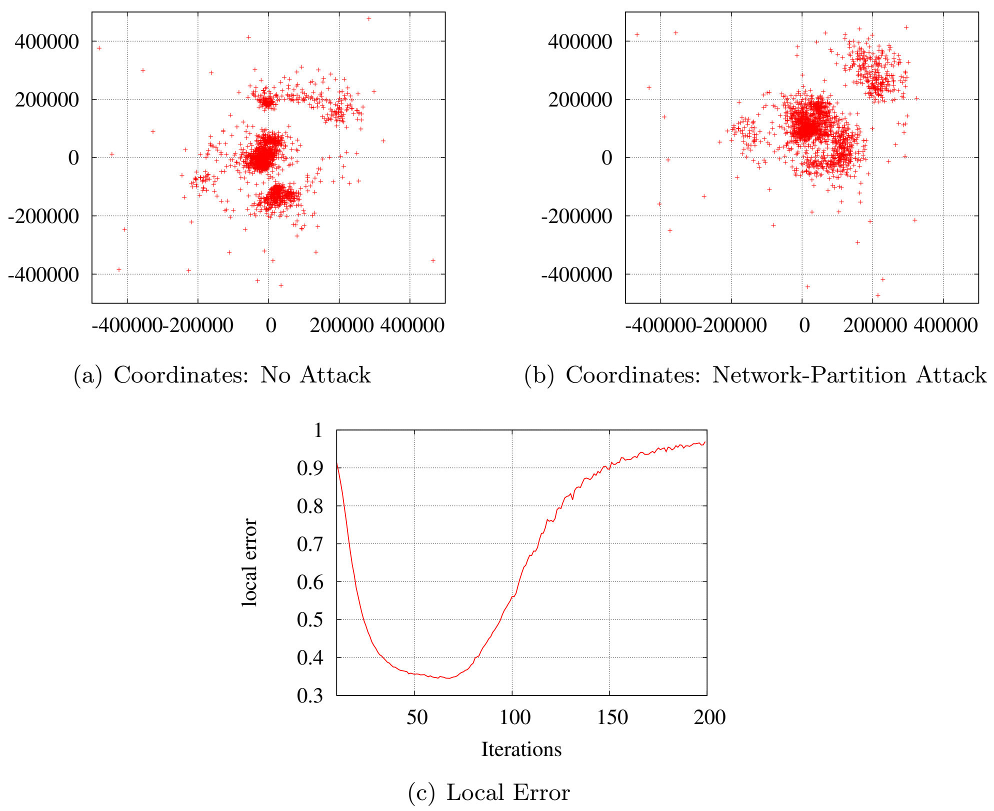
Figure 4.4: Impact of Frog-Boiling Attack

4.2 Defenses

Several works define security mechanisms to be able to detect and mitigate malicious behavior. The approaches for this differ in the way the mechanisms are structured. One can divide them in two different groups. The landmark-based defenses propose a subset of nodes to be landmarks, like a set of trusted nodes in order to secure virtual coordinate systems. The decentralized defenses do not have the need for a group of nodes to be considered as landmarks and work in a complete decentralized way.

4.2.1 Landmark-based Defenses

Kaafar *et al.* [54] show that the behavior of benign nodes can be modeled by a linear state space model, the coordinates can be seen as a stochastic process, as the RTT are statistical stable, and so the nodes need to update their coordinates only periodically. This behavior can be tracked by a Kalman filter. The Kalman filter is used to estimate Δ based on a set of measured relative errors, then the Kalman Filter gives the least mean squared estimates of Δ. Afterwards it gives an evaluation of that estimation through which outliers can be detected considering that there will be large deviations of the measured relative error from the mean value. So in order to detect malicious behavior they want to be sure that they do model the behavior of a benign node. Therefore, they make use of a set of trusted nodes, called surveyors. Those trusted nodes calculate their positions only using properties of other surveyors in order to calibrate the Kalman Filter. Once the surveyors have determined their



(a) Coordinates: No Attack (b) Coordinates: Network-Partition Attack

(c) Local Error

Figure 4.5: Impact of Network-Partition Attack

positions, other nodes in the network can track their behavior to know how normal behavior looks like and determine if other nodes behave maliciously.

Saucez *et al.* [88] define a reputation based system and use the notation of surveyors but in this case to model trust estimations for the other nodes. They introduce a further type of nodes the RCA - a certificating agent, that calculates a node's reputation.

Landmark-based defenses have stronger assumptions, as they require a predefined subset of nodes as a trusted node-set. Using landmarks as defense mechanisms decreases the scalability of virtual coordinate systems as well as trusted nodes to not provide a guarantee that no infiltration of attackers can happen. Therefore we focus in this work on decentralized defense mechanisms.

4.2.2 Decentralized Defenses

Veracity [94] is a decentralized VCS that introduces the notion of a verification set. Each node maintains a verification set where several other nodes attest to whether a particular update increases their estimation error above a certain threshold, and if so, ignores it. Nevertheless, how veracity performs if the verification sets are compromised is not clear.

Another approach is described by Wang *et al.* [109], where securing virtual coordinate systems is decoupled so that they consider securing the delay measurement and securing coordinates computation separately. For securing the delay measurements, [109] proposes to use a triangulation inequality violation (TIV) alert technique. TIV-alert uses the prediction ratio ($\frac{predicted}{measured}$) of coordinates to define that delays are falsified. Therefore they take the assumption that the prediction ratio is very small if the delay is forged. For securing the coordinates computation, [109] proposes to make use of the Byzantine Fault Detection (BFD) [46], scalable extension to Byzantine Fault Tolerance (BFT) [21] and uses PeerReview as accountability protocol. This approach has high cost for instance for the computation for public-key cryptography.

The outlier detection approach of Zage *et al.* [110, 111] does not rely on such assumptions, the overhead produced by outlier detection is low, and focuses on a decentralized VCS without any trusted components. The goal is to reduce the likelihood of a node to compute incorrect coordinates by ignoring malicious updates using statistical outlier detection. This outlier detection is done by each node before it updates its coordinates considering temporal and spatial correlations within the metrics in the network.

The outlier detection approach of [110, 111] does not need any big assumptions or a trusted node-set, we focus on this approach and explain the details in the following paragraphs:

Spatial Outlier Detection

The spatial outlier detection checks if the metrics reported by a node are consistent with the metrics of the neighbor nodes. This is done by receiving an observation tuple form a requested node. The observation tuple is related to the algorithm described in Algorithm 1 in Section 3.3 and consists of the three metrics: RTT, e_{remote}, Δ_{remote}. The last u updates are stored in a queue. As u is as large as the neighbor set, it is considered that every entry probably belongs to a different neighbor and one can calculate the centroid of the data contained in u. The Mahalanobis distance is calculated between the last observation tuple and the centroid. In order for the node to accept the new update, the computed distance must remain under a spatial threshold.

Temporal Outlier Detection

The temporal outlier detection checks the consistency of one node over time. Therefore the outlier detection takes into consideration the following 5-tuple: $\langle RTT$, e_{remote}, Δ_{remote}, e_{local}, $\Delta_{local}\rangle$. For each neighbor node a temporal centroid is computed using incremental learning. The newly received metrics are then compared to mean, standard deviation, and sample count for that node using a derived simplified Mahalanobis distance. The update is an outlier if it exceeds a temporal threshold.

Spatio-Temporal Outlier Detection

In [110, 111] the spatial outlier detection and the temporal outlier detection are combined. An update is only accepted if it does not exceed the temporal threshold as well as the spatial threshold. If an update is considered as an outlier, the update is ignored and not considered for the computation of the coordinates.

4.3 Conclusions

All the previous defenses have in common that their validation is made by running experiments and not by analyzing what defense mechanisms are optimal against what attack. The same, the proposed defense mechanisms, do not consider that an attacker can be aware of the defense in place. This assumption is unrealistic as attackers are often aware of the employed defense especially as design prinicples state that security should not rely on the secrecy of the defense algorithm [16]. Given that several attacks and several defense techniques exist, there is a need for a systematical evaluation identifying what are the best attack and defense strategies.

Furthermore, Chan-Tin *et al.* [22] show that all the previously defined defense mechanisms are vulnerable against the frog-boiling attack and the network-partition attack. Those two attackes are slow attacks, they remain undetected by defenses that model the behavior, as the attacks are imposed step-by-step, the model will learn these attacks as normal behavior. Furthermore, all threshold based defense mechanisms are vulnerable to those attacks, as the steps of the attack are under the threshold. An overview of the different defense mechansims is given in Table 4.1.

Frog-boiling attacks have been mitigated before in a different context by ANTIDOTE [87]. ANTIDOTE is a principal component analysis-based poisoning attack detector that constructs a low dimensional subspace which reflects most of the dispersion in the data. The required computations are relatively expensive and assumes an existing multidimensional input space. Such assumptions do not hold in our case, where we had first to map the one dimensional data to a higher dimensional space (which is the opposite of ANTIDOTE's subspace construction) and then rely on an efficient and online decision mechanism. A new defense mechanisms needs to address the frog-boiling attack in relation to virtual coordinate systems.

Table 4.1: Defenses - Overview

	Landmark based	Decentralized	Drawbacks	Experimental
Kaafar et al [54]	X	-	Trusted nodes can be compromised Vulnerable to Frog-Boiling	PlanetLab
Saucez et al [88]	X	-	Scalability	King data set
Veracity [94]	-	X	Trusted nodes can be compromised	PlanetLab Bamboo (King & Meridian data set)
Zage et al [111]	-	X	Vulnerable to Frog-Boiling	P2Psim (King, AMP, & Meridian data set)
Wang et al [109]	-	X	computational overhead	PlanetLab P2Psim (King data set)

Part II

Contributions

Chapter 5

Analyzing strategies for securing Virtual Coordinate Systems

In this chapter, we propose a game theoretical framework in order to analyze the adversarial behavior of an attacker and a defender. The goal of this work is to analyze the different defensive mechanisms in relation to the different attacking techniques. Therefore, we make use of Game Theory, as it provides the necessary mathematical techniques for analyzing strategic situations.

5.1 Introduction

We explore the development of a game theoretical framework with respect to Vivaldi, the Virtual Coordinate System, described in Chapter 3.3 and its attack and defense mechanisms described in Chapter 4. Previous works have proposed several ways to mitigate known attacks in VCS, but validation has always been done through experiments, with the assumption that the attacker is unaware of the defensive mechanism in place. However, practice has shown that attackers take the associated defense mechanism into account and try to bypass it. Hence, we investigate a scenario, where an attacker knows what defense is in place. We analyze the optimal defense strategy and the best attack strategy as there are several techniques available for both sides. This analysis is done with a systematical evaluation using a specifically described game model.

Furthermore, as a critical component of previous defense mechanisms is that a fixed threshold is used for deciding wether a data point is accepted by the system or suspected of coming from a malicious node, we leverage a framework with an adaptive threshold selection and test this framework against the defense mechanisms using fixed thresholds.

The contributions described in this Chapter are the following:

- We systematically study attack and defense techniques for virtual coordinate systems.
- We model rational attackers in virtual coordinate systems using the Nash equilibrium and irrational attackers using the quantal response equilibrium. From the defender side, we use game theory to tune our defensive mechanisms in order to mitigate the attacks.

- We define which attack has the biggest impact for large networks (*i.e.*, the King topology) and for smaller networks (*i.e.*, the AMP topology).

- We leverage an adaptive threshold selection and compare it to the strategies using fixed thresholds.

The structure of this chapter looks as follows, an introduction to game theory and its solution concepts is given in Section 5.2. We define the game model in Section 5.3, where we explore the motivation and the knowledge of each player. In Section 5.4, we elaborate the strategies and payoff functions for each player with respect to a basic game. Section 5.5 elaborates strategies and payoff functions for the players with respect to a more advanced game, where we integrate an adaptive threshold selection for the defense mechanisms. The evaluation of the different games is done in Section 5.6.

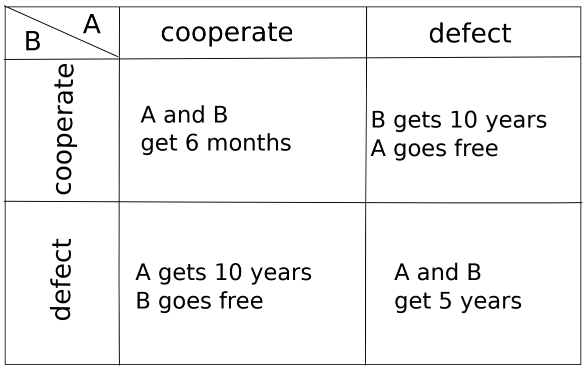
5.2 Background - Game Theory

Game theory was first introduced by John von Neumann in 1928 in his article "Zur Theorie der Gesellschaftspiele" [104], english translation: "On the Theory of Games of Strategy". In his work, Neumann defines a game for n players where every player wants to achieve a best possible gain, also known as payoff. The outcome of the game is defined as being dependent of the input of the players, the chosen strategies. John von Neumann worked on game theory related to the field of economics [74].

Although, game theory can be seen as something that happens all the time, mostly unconsciously by the players, as following Ken Binmore [15] - " A game is being played whenever people have anything to do with each other". Game theory is useful in any strategic situations with n players, where every player has its own strategy set and for every chosen action the players receive a payoff in form of a gain or punishment. Therefore it is not surprising that it became a relevant topic in network security [66, 69, 85, 100, 106, 108]. One of the first approaches for applying game theory to network security is described by McInerney *et al.* [69]. In this work, an underlying Markovian decision process and a simple one-player game are used to reason, detect, and respond to automated attack behavior in information assurance systems. Similar work on network security by Lye and Wing [66] models the interaction between an attacker and a defender as a two-player stochastic game. The explicit enumeration of states as described in the previous two papers is impossible in our context due to the large number of attacking nodes that we consider. Game theoretical concepts have also been applied to model attackers and high-interaction honeypots [106] for configuring reciprocal actions. In [108], the authors propose to break the anonymity of Tor [35, 101], and provide countermeasures that are modeled using game theory. Roy et al. [86] give an overview of the different works on applying game theory to network security.

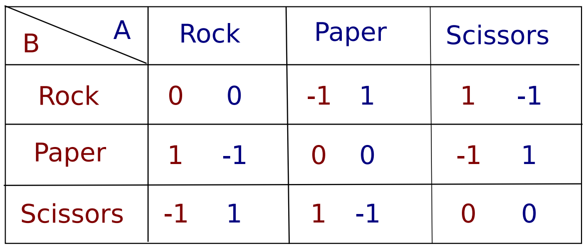
Some of our previous work has used game theory to understand and better defend against blocking and flooding attacks within distributed hash tables used in P2P Session Initiation Protocol infrastructures [14].

Once, the game model is defined, including the strategy sets and the payoff functions, we need to find a solution for predicting how the game will be played. Consequentially, we will be able to conduct experimental analysis. In this work, we make use of the well known Nash Equilibrium and on the Quantal Response Equilibrium. In the following sections, we explain the two equilibria.



B \ A	cooperate	defect
cooperate	A and B get 6 months	B gets 10 years A goes free
defect	A gets 10 years B goes free	A and B get 5 years

Figure 5.1: Prisonner's Dilemma - example for pure strategy



B \ A	Rock	Paper	Scissors
Rock	0 0	-1 1	1 -1
Paper	1 -1	0 0	-1 1
Scissors	-1 1	1 -1	0 0

Figure 5.2: Rock, Scissors, Paper - example for mixed strategy

5.2.1 Nash Equilibrium

The first reason why using the Nash Equilibrium [70], is that it introduces and defines methods for analyzing non-cooperative games, where the players do not cooperate or communicate. This exactly represents our game model, as we have two opponents with conflictive goals. The second reason is that the Nash Equilibrium defines that the players are rational. That means that the players will not differ from the outcome of the Nash Equilibrium, as they will not gain greater payoff with any other action strategy. In this model, we assume that an attacker wants to harm the network as much as possible and the defender wants do defend the system as good as possible, these statements are reflected in the payoff functions. Concluding, a player has no reason to deviate from the outcome, a strategy, of the Nash Equilibrium as this is the most they can achieve.

Calculating the Nash equilibrium can result in a pure strategy, where it is defined that a player, attacker or defender, plays constantly the same action out of the corresponding action set (A or D). To illustrate a pure strategy, we consider the well known prisoner's dilemma [103]. The prisoner's dilemma is about two people A and B that have been arrested. It is assumed that they committed a crime. They are placed in separate cells and both have the option to either provide evidence to incriminate the other or to be silent and cooperate with the partner. If only one chooses to provide evidence against the other (defecting on his partner) than that one will gain freedom while the partner gets sentenced to 10 years in prison. If both provide evidence, than both will get 5 years in prison. If the two are cooperating with each other then both well get 6 months in prison. The strategies and the payoffs are represented in Figure 5.2.1. As the prisoners are separated, they cannot know what the other one is choosing. Because of that, the best choice following the Nash equilibrium is to defect for both prisoners as they might risk to get a much higher prison sentence when one is cooperating while the other is defecting. This is a pure strategy.

A Nash equilibrium can also result in a mixed Strategy S_i. A mixed strategy is a probabilistic distribution over the corresponding pure strategies, $S_i : D \to [0,1]$ and $S_{-i} : A \to [0,1]$. The player will randomly play each strategy while each strategy is selected with the associated probability. To illustrate a mixed strategy we consider the popular hand game *Rock, Paper, Scissors* [38] often used as a choosing game like flipping a coin. Consider two players A and B, their strategies are composed of either playing *Rock*, *Scissors*, or *Paper*. Rock breaks scissors. Scissors cut paper and paper covers rock. The payoffs are represented in Figure 5.2.1. There is no pure strategy, as it is unknown what the other one is playing. In this case, it is the best for the players to play *Rock* with the probability of 0.33, *Scissors* with the probability of 0.33, and *Paper* as well with a probability of 0.33. This is a mixed strategy.

The mixed strategy profile S_i is a Nash equilibrium if for each player i there is no other strategy profile S'_i that will lead to a higher gain with respect to the strategy profile S_{-i} applied by the opponent $-i$, meaning that for player i there is no average payoff Q_i greater than the one for the strategy profiles $S_i, S_{-i} : Q_i(S_i, S_{-i}) \geq Q_i(S'_i, S_{-i})$ where

$$Q_i(S_i, S_{-i}) = \sum_{q=1}^{n} S_i(d_q) \sum_{p=1}^{k} S_{-i}(a_p) * M_i(d_q, a_p)$$

In this equation M_i represents the payoff matrix of player i and $M_i(d_q, a_p)$ is the payoff for player i while choosing d_q as action while the opponent plays a_p.

5.2.2 Quantal Response Equilibrium

The Quantal Response Equilibrium (QRE) [43] is a solution concept that generalizes the Nash equilibrium by introducing an error parameter to the payoff function. This is motivated by the fact that payoff functions may be erroneous and one can not have total certainty about the payoff value. In this manner, the regular QRE provides an equilibrium with bounded rationality in contrast to the Nash equilibrium which defines all the players to be completely rational.

The error parameter λ, also called the rationality parameter, is varied until the regular QRE converges to the Nash equilibrium. Rationality in this sense means that no player is motivated to diverge from the Nash Equilibrium as there is no other strategy where one can gain more than the ones specified in the resulting strategy profile of the Nash Equilibrium. On the opposite, irrationality means that even though the attacker can not gain greater payoff, he will chose another strategy than the one defined by the Nash Equilibrium. When $\lambda = 0$, the player is completely irrational, in this case he could, for instance, chose the strategy randomly and when $\lambda \to \infty$, the player becomes perfectly rational and follows the Nash equilibrium. We calculate the QRE by using the following equation which defines the probability of player i to choose strategy d_q out of action set D:

$$S_{id_q} = \frac{\exp \lambda \times U_{i(d_q)}(S_{-i})}{\sum_{a_p} \exp^{\lambda \times U_{i(a_p)}(S_{-i})}}$$

where $U_{i(d_q)}(S_{-i})$ describes the expected utility for player i using strategy d_q while considering other players to play with a probability distribution S_{-i}: $U_{i(d_q)} = \sum_{p=1}^{k} S_{-i}(a_p) * M(d_q, a_p)$.

We use the QRE to quantify how irrational a player can be while still maintaining the same equilibrium profiles.

5.3 Defining the Game Model

A strategic situation is composed of two or more players, all of them have their own strategies and motivations. By playing a specified strategy a player receives a reward, depending on the strategy chosen by other players.

Our game model consists of two players, the defender i and the attacker $-i$. Both players are opponents and their goals and motivations are conflictive. Each player has its own strategy set: A for the attacker and D for the defender. A strategy set is composed of a given number of actions that a player can choose to play. An action can be considered as a synonym for strategy. For every chosen and played action, each player receives a payoff P. This payoff describes the effectiveness of the chosen action with respect to the chosen action of the opponent and is based on the payoff function of the player.

We consider a static or strategic game. This is a one-shot game and means that the decisions of the two players are taken simultaneously and unknown to the opponent. In order to leverage an analytical game model, first the aim of the players and the knowledge of each opponent is defined. Second, the strategies for each player are explored. Third, we define the payoff functions. Those payoff functions represent the aim of each player and are used to statistically analyze the different strategies.

5.3.1 Adversarial Motivations and Knowledge

In this context, we assume that an attacker wants to harm the system as much as possible by interfering with the latency estimations that are crucial for the well-functioning of Vivaldi. Meaning that an attacker wants to modify latencies between nodes so that benign nodes are either updating their coordinates to incorrect positions or prevented from updating to the correct coordinates. This leads to a decreasing of the accuracy of the system as described in Section 4.1. Of course, the attacker wants to have as much impact as possible on the system, which includes that he does not want that his attack gets detected and mitigated, as this will lead to decrease the impact of its attack.

On the opposite site, we assume that the defender wants to have a good accuracy, as the nodes should find soon their correct positions and remain there. Accordingly, the defender wants the attacks to have least impact as possible, meaning that the attack should be detected and avoided by not accepting updates from a malicious node.

We assume that the players are aware of the opponent's strategy set. However, our game is considered as a silent game, as the opponents do not know which action out of the strategy set was chosen in the past. The same, the players do not know what strategy the opponent will choose for a current game.

5.4 Determining the Basic Game

In this section, we define the strategies and payoff functions for the basic game. We call this game basic, as it includes the basic strategies of the defender. These basic strategies are based on the defense mechanisms defined in the previous work [111]. The defender's and the attacker's strategies are explored in the following section as well as the respective payoff functions.

5.4.1 Strategies

The attacker can conduct three different types of attacks, namely the inflation attack, the deflation attack, and the oscillation attack. Additionally, the attacker is able to modify the amount of attacking nodes between 10%, 20%, and 30%. Hence the action set, or strategy set, of the attacker is as follows:

A : {inflation - 10% attackers; deflation - 10% attackers; oscillation - 10% attackers;
inflation - 20% attackers; deflation - 20% attackers; oscillation - 20% attackers;
inflation - 30% attackers; deflation - 30% attackers; oscillation - 30% attackers}

On the opposite side, the defender can use three different defense mechanisms, the spatial outlier detection, the temporal outlier detection, and the combination of both, the spatio-temporal outlier detection. All those mechanisms use a fixed threshold value in order to define if a new update is an outlier or not (see 4.2). The action set of the defender looks as follows:

D : {spatial outlier detection; temporal outlier detection; spatio-temporal outlier detection}

5.4.2 Payoff Functions

The objective of payoff functions is to reflect the players' motivations and intents. The motivations of the opponents are described in 5.3.1. Following these descriptions, we notice that both players want to influence the accuracy of the system, so this shows that the accuracy has to be reflected in both payoff functions. The accuracy of Vivaldi is reflected in the system's error values.

- The system prediction error $Error_{pred}$ measures the accuracy of the overall virtual coordinate system by calculating the difference between the real, measured round-trip-time and the estimated round-trip-time.
- The relative error $Error_{rel}$ quantitatively compares the effect of the adversaries on the accuracy of the virtual coordinate system by calculating the relation between the system prediction error under attack and the system prediction error when no attack takes place.

The attacker wants to decrease the accuracy of the system. Additionally, having a greater amount of nodes to perform an attack, has a higher cost as consequence for the attacker, as it will take an higher effort to compromise and coordinate the nodes. Following the motivation of the attacker, the payoff will increase if the system error values increase, so we deduce following payoff functions for the attacker:

- $P_{attacker_{pred}} = \frac{Error_{pred}}{\%\text{ attackers}}$
- $P_{attacker_{rel}} = \frac{Error_{rel}}{\%\text{ attackers}}$

The defender has the opposite goal, he wants to increase the accuracy of the system. Accordingly, his payoff increases, when the system error values decrease, the payoff functions for the defender look as follows:

- $P_{defender_{pred}} = -Error_{pred}$
- $P_{defender_{rel}} = \frac{1}{Error_{rel}}$

We just defined the payoff function for the two opponents, but we want to integrate the goals of the two players in a more specific way. Therefore we define two different games (Game 1 and Game 2) based on different percentiles of the system error values representing different motivations.

Game 1

As already known, the attacker wants to harm the system as much as possible. In this game, we represent this goal, by taking the 5^{th} percentile of the system error values as payoff function. This way, if the attacks results in a high 5^{th} percentile, this means that even the 5% lowest values are high, so concluding the remaining 95% are even higher. So the attacker's payoff functions become:

- $P_{attacker\_5^{th}pred} = \frac{5^{th}Error_{pred}}{\text{\% attackers}}$
- $P_{attacker\_5^{th}rel} = \frac{5^{th}Error_{rel}}{\text{\% attackers}}$

On the contrary, the defender wants to protect as many nodes as possible and so wants to have a low 95^{th} percentile of the system error value. Accordingly, the defender's payoff functions become:

- $P_{defender\_95^{th}pred} = -95^{th}Error_{pred}$
- $P_{defender\_95^{th}rel} = \frac{1}{95^{th}Error_{rel}}$

Game 2

In this game, we consider an average overview of the strategic situation of the system and use therefore the median system error values, so the payoffs become:

- $P_{attacker\_50^{th}pred} = \frac{50^{th}Error_{pred}}{\text{\% attackers}}$
- $P_{attacker\_50^{th}rel} = \frac{50^{th}Error_{rel}}{\text{\% attackers}}$
- $P_{defender\_50^{th}pred} = -50^{th}Error_{pred}$
- $P_{defender\_50^{th}rel} = \frac{1}{50^{th}Error_{rel}}$

We summarize the different strategies and payoff function for Game 1 and Game 2 in Table 5.1.

Table 5.1: Overview of the Basic Games: Game 1 and Game 2

Player	Strategy		Payoff function	
			border value	median value
Defender	Spatial		$P_{def\_95^{th}pred} = -95^{th}Error_{pred}$	$P_{def\_50^{th}pred} = -50^{th}Error_{pred}$
	Temporal		$P_{def\_95^{th}rel} = \frac{1}{95^{th}Error_{rel}}$	$P_{def\_50^{th}rel} = \frac{1}{50^{th}Error_{rel}}$
	Spatial-Temporal			
Attacker	10%	Inflation Deflation Oscillation	$P_{att\_5^{th}pred} = \frac{5^{th}Error_{pred}}{\%\text{ mal. nodes}}$	$P_{att\_50^{th}pred} = \frac{50^{th}Error_{pred}}{\%\text{ mal. nodes}}$
	20%	Inflation Deflation Oscillation		
	30%	Inflation Deflation Oscillation	$P_{att\_5^{th}rel} = \frac{5^{th}Error_{rel}}{\%\text{ mal. nodes}}$	$P_{att\_50^{th}rel} = \frac{50^{th}Error_{rel}}{\%\text{ mal. nodes}}$

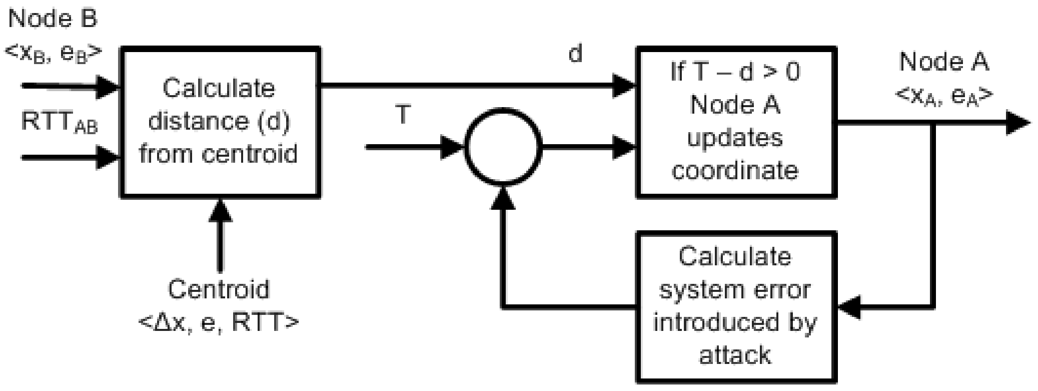

Node B
<x_B, e_B>
RTT_AB
Calculate distance (d) from centroid
Centroid
<Δx, e, RTT>
d
T
If T – d > 0 Node A updates coordinate
Node A
<x_A, e_A>
Calculate system error introduced by attack

Figure 5.3: Integration of control theory - Block Diagram

5.5 Determining the Advanced Game

In this game, we leverage new strategies for the defender. We focus on optimizing the threshold selection of the spatial outlier detection. Usually, fixed thresholds are applied that have been determined experimentally as in [111]. Such an approach is inflexible, prone to errors, and may be exploited by an adversary to remain undetected. Therefore, we integrate a feedback control loop in order to update the threshold on the run.

5.5.1 Adaptive Threshold Selection

Below, we show how by leveraging control theory [34, 37] we design an adaptive threshold technique to improve the threshold selection.

Figure 5.3 shows how the adaptive threshold selection is integrated with the outlier detection mechanism. A feedback control loop is regularly updating the spatial threshold with the objective to tighten the threshold and adapt to attacks. The update of the threshold is based on the observation that more severe attacks (as per the nature of the attack and percentage of malicious nodes) will result in higher differences between the estimated RTT (predicted by the coordinates resulted from the virtual coordinate system) and the actual RTT. Specifically, at time n, the new threshold T_n is updated based on the threshold at time $n-1$, T_{n-1}, and the difference between the current prediction error $Error_{attack}(n)$ and an ideal value for the prediction error $Error_{no\_attack}(n)$ as follows:

$$T_n = T_{n-1} - c\frac{(Error_{attack}(n) - Error_{no\_attack}(n))}{RTT_{Est}(n)} \tag{5.1}$$

where c is a constant to define the importance of the prediction error $Error_{attack}(n)$ and $RTT_{Est}(n)$ is the current estimated RTT. The prediction error $Error_{attack}(n)$ is based on all the prediction errors calculated during each update of each single node, one can either take an averaged value or percentile values. The average value is more likely to be affected by potential malicious values that bypassed the outlier detection. Thus, different defense strategies in the case of the adaptive threshold selection involve using different percentile values in the prediction error.

We take into account that the prediction error varies over time, as before the system stabilizes nodes have high prediction error and must update their coordinates by large amounts. However, after some iterations the nodes converge to their correct coordinates which results in low prediction errors. To define an independent evaluation of the ideal value, we ran several experimental runs without the presence of attacks and without outlier detection.

In summary, the effectiveness of an outlier detection technique that uses the adaptive threshold selection depends on the value of c, which defines the importance of the prediction error, and the percentile values on which the prediction error used in updating the threshold is computed. Figure 5.4 shows the impact of the different values for c, whereas $c = 0.0$ is the case where the control theory is inactive, as the threshold will remain stable, so this can be seen as the normal spatial outlier detection. We see that the curves for $c <> 0.0$ all perform better than for $c = 0.0$. This shows that control theory improves the spatial outlier detection.

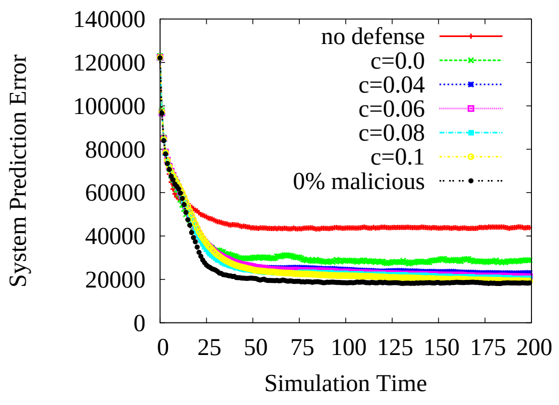
In Figure 5.5, we observe the decreasing threshold while using different values for c. We see that with an increasing amount of malicious nodes - 30% attackers in Figure 5.5(c), the threshold decreases faster than for 10% attacker (Figure 5.5(a)).

5.5.2 Strategies

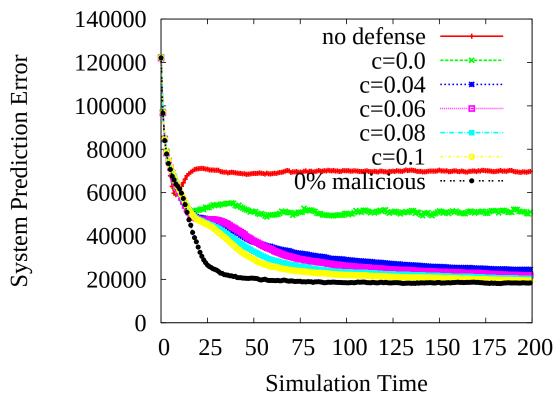
The strategies for the attacker are the same as in the previously elaborated basic game, so he can conduct three different types of attacks, inflation, deflation, and oscillation attack with three different amounts of attacking nodes, 10%, 20% or 30% malicious nodes. The strategy set of the attacker is:

A : {inflation - 10% attackers; deflation - 10% attackers; oscillation - 10% attackers;
inflation - 20% attackers; deflation - 20% attackers; oscillation - 20% attackers;
inflation - 30% attackers; deflation - 30% attackers; oscillation - 30% attackers}

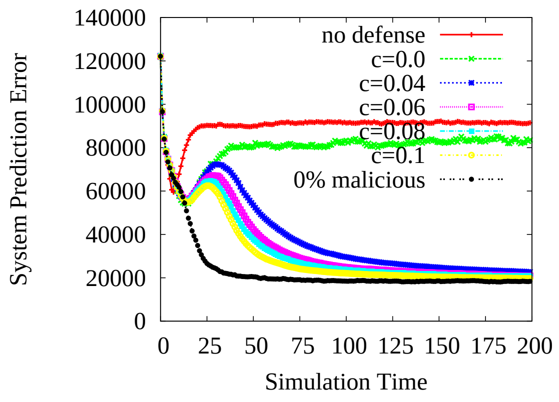
The defender's strategy set is composed of the selection of $c : \{0; 0.04; 0.06; 0.08; 0.1\}$ of the control loop and the percentiles $\{25^{th}; 50^{th}; 75^{th}\}$ of the prediction error used in Equation 5.1. We include $c = 0$ in this strategy set, in order to compare the usage of the control loop to the fixed threshold



(a) Inflation Attack - 10% attackers



(b) Inflation Attack - 20% attackers



(c) Inflation Attack - 30% attackers

Figure 5.4: Impact of different values for c using the median prediction error - p2psim - 1740 nodes

selection, as the threshold will not be changed when $c = 0$. The strategy set for the defender looks as follows:

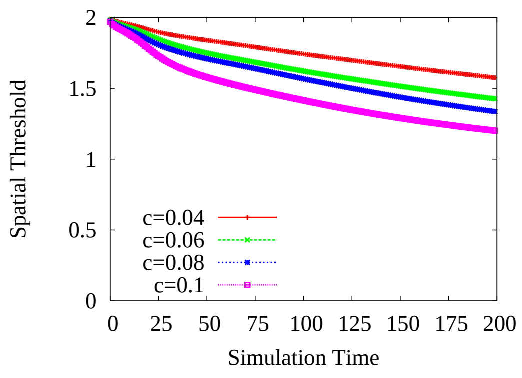
$$D : \{c = 0; c = 0.04, 25^{th}perc; c = 0.06, 25^{th}perc; c = 0.08, 25^{th}perc; c = 0.1, 25^{th}perc;$$
$$c = 0.04, 50^{th}perc; c = 0.06, 50^{th}perc; c = 0.08, 50^{th}perc; c = 0.1, 50^{th}perc;$$
$$c = 0.04, 75^{th}perc; c = 0.06, 75^{th}perc; c = 0.08, 75^{th}perc; c = 0.1, 75^{th}perc\}$$

5.5.3 Payoff Functions

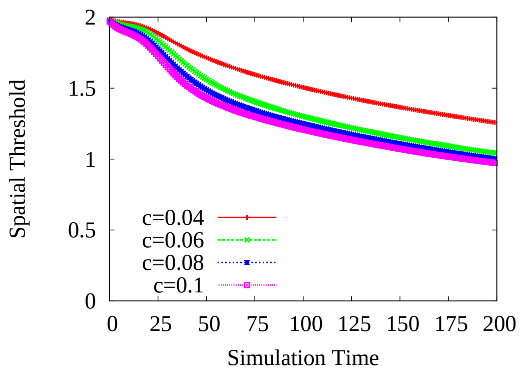
We define two different payoff functions for the attacker, these functions represent a different aspect of the attacker's motivation. We know that an attacker wants to harm the system and this is reflected in the system prediction error, which gives an overview of the accuracy of the system. The first payoff function for the attacker looks as follows:

- $P_{attacker_{pred}} = Error_{pred}$

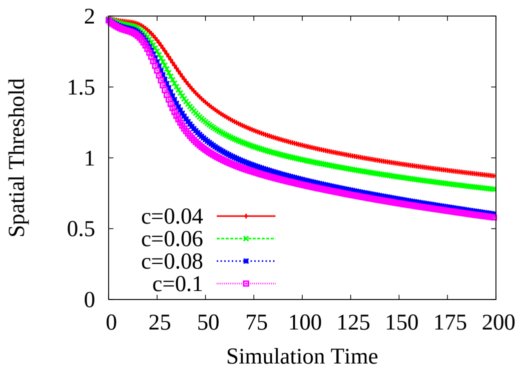
The second payoff function reflects the need for the attacker to remain undected. We assume that an attacker is aware of the defense mechanisms, although he does not know the details, like selected



(a) Inflation Attack - 10% attackers



(b) Inflation Attack - 20% attackers



(c) Inflation Attack - 30% attackers

Figure 5.5: Threshold behavior for different values for c using the median prediction error - p2psim - 1740 nodes

values, but he is aware of the feedback control loop. That leads to the aim of the attacker to keep an high threshold value, so that it is easier for him to remain under the radar. So we take the averaged threshold T_{avg} as payoff for the attacker. The bigger T_{avg} is, the bigger the gain for the attacker.

- $P_{attacker_T} = T_{avg}$

The defender wants to have an high accuracy of the system, which means that he wants to have a small system prediction error, therefore the payoff function for the defender is:

- $P_{defender} = -Error_{pred}$

An overview of the advanced game is given in Table 5.2.

5.6 Experimental Results

In this section, we demonstrate, through simulations using actual Internet topologies and quantitative analysis using game theory techniques, the efficacy of different attacks at impacting the accuracy of the

Table 5.2: Overview of the Advanced Game

Player	Strategy		Payoff function	
			Error evaluation	Threshold evaluation
Defender	$c = 0$		$P_{def} = -Error_{pred}$	
	25th Percentile	$c = (0.04, 0.06, 0.08, 0.1)$		
	50th Percentile	$c = (0.04, 0.06, 0.08, 0.1)$		
	75th Percentile	$c = (0.04, 0.06, 0.08, 0.1)$		
Attacker	10%	(Inflation, Deflation, Oscillation)	$P_{att} = Error_{pred}$	$P_{att} = T_{avg}$
	20%	(Inflation, Deflation, Oscillation)		
	30%	(Inflation, Deflation, Oscillation)		

Vivaldi virtual coordinate system and of our defense mechanisms at preserving its ability to maintain accurate latency estimates.

In order to simulate the attack and defense strategies, we use the King [45] and AMP [4] data sets in conjunction with the p2psim simulator [77]. The King data set contains the pair-wise RTT of 1740 nodes with an average RTT of 180ms and was selected since it is representative of larger scale peer-to-peer systems and has been used in validating several virtual coordinate systems. The AMP data set consists of the pair-wise RTT of 90 nodes with an average RTT of 70ms and it is used to represent smaller, high speed systems (*e.g.*, a corporate network). Synthetic topologies are not considered as they do not capture important network properties inherent in real networks such as violations of the triangle inequality.

We ran simulations for each combination of attack type and defense strategy described in Chapter 4. We ran each simulation for 200 time units, where each time unit is 500 seconds in length. Every simulation was run ten times with the reported metrics averaged over all of the runs. The nodes join in a flash-crowd scenario in which all nodes join simultaneously and are initially placed at the origin of the logical coordinate space. All nodes that join the network are physically stationary and are present for the duration of the experiment. Each node proceeds independently of other nodes and chooses a reference set of 64 nodes using the Vivaldi method where half of the nodes are selected as the closest nodes based on network latency and the rest are selected at random. All other Vivaldi parameters were initialized to the optimal values discussed by Dabek *et al.* [32].

5.6.1 Basic Game

Table 5.3: King - Equilibrium Points Based on *Game 1*

Threshold	Error metric	Nash equilibrium strategy profile		
		profile	attacker	defender
1.25	pred. error	pure	Infl/10% att.	Spatial-temporal
	rel. error	pure	Infl/10% att.	Spatial-temporal
1.5	pred. error	pure	Infl/10% att.	Spatial-temporal
	rel. error	pure	Infl/10% att.	Spatial-temporal
1.75	pred. error	pure	Infl/30% att.	Spatial
	rel. error	2 pure profiles	Infl/30% att. Infl/30% att.	Spatial-temporal Spatial

We first analyze the effect of using different spatial outlier thresholds on the Vivaldi virtual coordinate system running over the King topology. In Table 5.3, we can see the Nash equilibrium using *Game 1* as defined in Table 5.1. From the results, we see that the inflation attack has a large impact on the system. Under this attack, we find that the most efficient defense strategy is spatial-temporal outlier detection when using lower spatial outlier thresholds (*e.g.*, ≤ 1.5). For higher thresholds, both spatial-temporal outlier detection and spatial outlier detection provide similar defense performance. We note that temporal outlier detection is ineffective as it never appears as part of one of the equilibria.

Table 5.4: King - Equilibrium Points Based on *Game 2*

Threshold	Error metric	Resulting Nash equilibrium strategy profile		
		profile	attacker	defender
1.25	pred. error	pure	Infl/10% att.	Spatial
	rel. error	pure	Infl/10% att.	Spatial
1.5	pre. error	pure	Infl/10% att.	Spatial
	rel. error	pure	Infl/10% att.	Spatial
1.75	pred. error	pure	Infl/30% att.	Spatial-temporal
	rel. error	pure	Infl/30% att. Infl/30% att.	Spatial-temporal Spatial

In Table 5.4, we present the Nash equilibrium using *Game 2*. Depending on the threshold selected, either the spatial outlier detection or the spatial-temporal outlier detection defense techniques provide the best performance and are thus employed in the resulting Nash equilibrium. Based on the evaluations of both *Game 1* and *Game 2*, we conclude that the inflation attack has the greatest potential to impact the virtual coordinate system. It is interesting to note that for lower outlier thresholds (≤ 1.5), the attack is most effective for smaller percentages of malicious nodes as the effort to create larger attacks leads to diminishing returns. Only the higher threshold of 1.75 allows the inflation attack with 30% malicious nodes to be effective and appear as an equilibrium, allowing us to conclude this threshold is less effective at mitigating the effects of the attacker. Finally, similar to *Game 1*, we notice that temporal outlier detection does not appear in an equilibrium and we thus conclude that this type of outlier detection is not an effective countermeasure when used by itself.

We also analyze the best defenses against the different attacks when using a spatial outlier threshold of 1.5, as this value was suggested by previous research [111]. For the deflation attack, the optimal defense strategy is to use spatial outlier detection as it results in a pure equilibrium for both the prediction error and the relative error. The spatial-temporal outlier detection is the best defensive mechanism against the oscillation attack regarding both error metrics. Evaluations based on *Game 1* show that spatial outlier detection performs similarly. For the inflation attack, we have a different defense strategy resulting in a pure equilibrium for each of the games. For *Game 1*, spatial-temporal outlier detection represents the pure equilibrium, while in *Game 2*, spatial outlier detection represents the equilibrium. Furthermore, we assess the threshold selection for this data set. we present the resulting Nash equilibria Table 5.5, Table 5.6 and Table 5.7 show the resulting Nash equilibria for choosing the best spatial threshold value. We find independent of the game or the error metric, that a threshold value of 1.25 always results in a pure equilibrium, making this the best threshold. Table 5.8 and Table 5.9 present the best defense mechanism with respect to the specific attack mechanism.

The previous results, which are based on the Nash equilibrium, assume that the players are completely rational. As this cannot be guaranteed, we use a secondary evaluation to determine how irrational the players can act while still maintaining the same optimal equilibrium profiles. In Figure 5.6, we present the regular QRE for the data set. The y-axis represents the probability for a

Table 5.5: King - Equilibrium Points for Threshold Selection

Game	Error metric	Nash equilibrium profile		
		profile	attacker	defender
Game 1	pred. Error	pure	Infl/10% att.	T1.25/Spatio-temporal
	rel. Error	pure	Infl/10% att.	T1.25/Spatio-temporal
Game 2	pred. Error	pure	Infl/10% att.	T1.25/Spatial
	rel. Error	pure	Infl/10% att.	T1.25/Spatial

Table 5.6: AMP - Equilibrium Points for Threshold Selection based on *Game 1*

Error metric	Nash equilibrium profile				
	profile	attacker		defender	
		strategy	probability	strategy	probability
pred. Error	3 mixed equilibrium points	Infl/10% att.	0.25	T1.25/Spatio-temporal	0.51
		Osci/10% att.	0.75	T1.5/Spatial	0.49
		Infl/10% att.	0.14	T1.25/Spatio-temporal	0.58
		Defl/10% att.	0.26	T1.5/Spatial	0.32
		Osci/10% att.	0.60	T1.75/Spatial	0.10
		Infl/10% att.	0.16	T1.25/Spatio-temporal	0.63
		Defl/10% att.	0.52	T1.25/Temporal	0.04
		Osci/10% att.	0.32	T1.75/Spatial	0.33
rel. Error	mixed	Infl/10% att.	0.18	T1.25/Spatio-temporal	0.67
		Defl/10% att.	0.51	T1.25/Temporal	0.07
		Osci/10% att.	0.31	T1.5/Spatio-temporal	0.26

Table 5.7: AMP - Equilibrium Points for Threshold Selection based on *Game 2*

Error metric	Nash equilibrium profile				
	profile	attacker		defender	
		strategy	probability	strategy	probability
pred. Error	mixed	Infl/10% att.	0.31	T1.25/Spatio-temporal	0.42
		Defl/10% att.	0.48	T1.25/Spatial	0.24
		Osci/10% att.	0.21	T1.5/Spatial	0.34
rel. Error	mixed	Infl/10% att.	0.31	T1.25/Spatio-temporal	0.43
		Defl/10% att.	0.47	T1.25/Spatial	0.21
		Osci/10% att.	0.22	T1.5/Spatial	0.36

strategy for a given λ. We notice that when considering the prediction error (Figure 5.6(a)), the QRE converges to the Nash equilibrium for $\lambda \to 0$, implying that even if the attacker is irrational, he will follow the Nash equilibrium with respect to the prediction error. Regarding the relative error, the QRE converges to the Nash equilibrium for $\lambda \approx 300$(Figure 5.6(b)) which means that the strategies in relation to the relative errors also converge fast to the Nash Equilibrium as $0 < \lambda < \infty$. Using this metric as the basis of the payoff function, an irrational attacker will diverge from the Nash Equilibrium, but as it becomes more rational, it quickly follows the optimal identified strategy.

We evaluate the AMP data set looking at both error metrics for different spatial outlier threshold selections. Table 5.10 describes the resulting strategy profiles from following *Game 1*. We notice that for this data set, the resulting strategy profiles are not nearly as homogeneous as those for the King data set. Most of the resulting strategy profiles consist of a mixed strategy, meaning that the different

Table 5.8: Equilibrium Points against specific Attacks based on *Game 1*

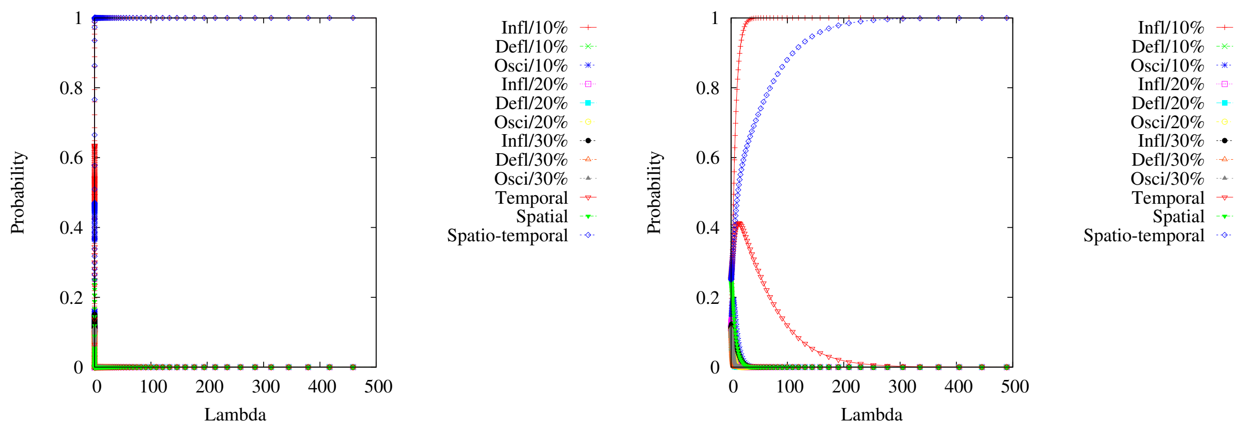
Attack	Data Set	Error metric	Nash equilibrium profile		
			profile	attacker	defender
Inflation	King	pred.	pure	10% att.	Spatio-temporal
		rel.	pure	10% att.	Spatio-temporal
	AMP	pred.	pure	10% att.	Spatial
		rel.	pure	10% att.	Spatial
Deflation	King	pred.	pure	10% att.	Spatial
		rel.	pure	10% att.	Spatial
	AMP	pred.	pure	10% att.	no defense
		rel.	pure	10% att.	no defense
Oscillation	King	pred.	pure	10% att.	Spatio-temporal
		rel.	2 pure equilibria	10% att.	Spatial
				10% att.	Spatio-temporal
	AMP	pred.	pure	10% att.	Spatial
		rel.	pure	10% att.	Spatial

Table 5.9: Equilibrium Points against specific Attacks based on *Game 2*

Attack	Data Set	Error metric	Nash equilibrium profile		
			profile	attacker	defender
Inflation	King	pred.	pure	10% att.	Spatial
		rel.	pure	10% att.	Spatial
	AMP	pred.	pure	10% att.	Spatial
		rel.	pure	10% att.	Spatial
Deflation	King	pred.	pure	10% att.	Spatial
		rel.	pure	10% att.	Spatial
	AMP	pred.	pure	10% att.	Spatial
		rel.	pure	10% att.	Spatial
Oscillation	King	pred.	pure	10% att.	Spatio-temporal
		rel.	pure	10% att.	Spatio-temporal
	AMP	pred.	pure	10% att.	Spatial
		rel.	pure	10% att.	Spatial

strategies should be utilized with the given probability in order to be as effective as possible. For example, given the spatial outlier threshold of 1.25, the attacker has the most impact while applying the deflation attack with only 10% malicious nodes in the system with a probability of 0.55 and applying the oscillation attack with 10% malicious nodes in the system with a probability of 0.45. The countermeasures look similar, applying spatial-temporal outlier detection and temporal outlier detection with probabilities of 0.93 and 0.07 respectively. Overall, we can see that the spatial-temporal outlier detection has highest probability of being applied. Interestingly, unlike the King data set, the temporal outlier detection is often part of the equilibrium, but only with low probability. The outcomes for *Game 2* are reflected in Table 5.11. For this evaluation, only the spatial-temporal outlier detection and the spatial outlier detection are considered in the equilibriums.

Next, we investigate the optimal countermeasure with respect to the different attacks. The spatial outlier detection performs best against the three attacks. Assessing the different thresholds, results show that a threshold value of 1.25 is the best choice, a threshold value of 1.5 is second best, 1.75



(a) Prediction error (b) Relative error

Figure 5.6: The Quantal Response Equilibrium evaluation for the King data set based on *Game 1*

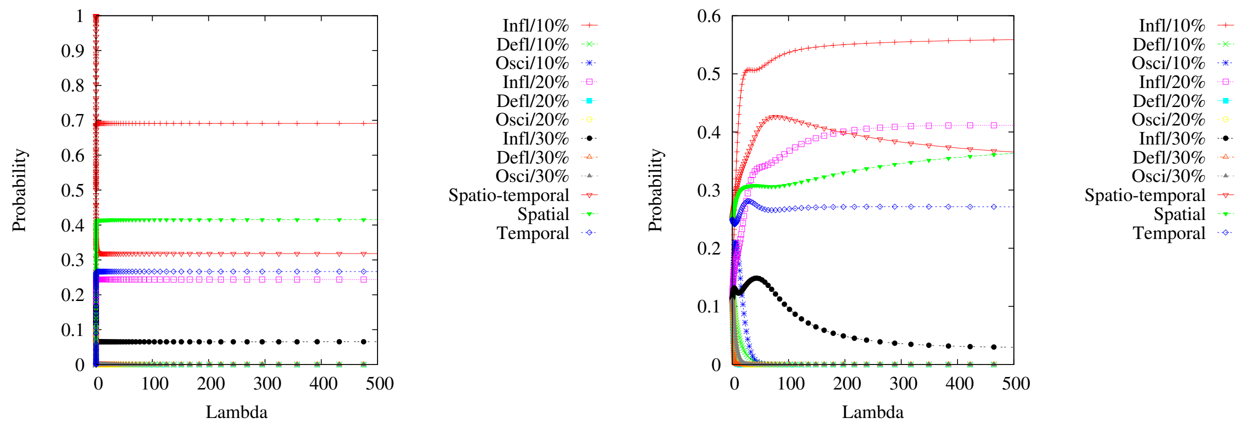
Table 5.10: AMP - Equilibrium Points Based on *Game 1*

Threshold	Error metric	Nash equilibrium profile				
		profile	attacker		defender	
			strategy	probability	strategy	probability
1.25	pred.	mixed	Defl/10% att.	0.55	Spatial-temporal	0.93
			Osci/10% att.	0.45	Temporal	0.07
	rel.	mixed	Defl/10% att.	0.54	Spatial-temporal	0.92
			Osci/10% att.	0.46	Temporal	0.08
1.5	pred.	pure	Infl/10% att.		Spatial	
	rel.	pure	Infl/10% att.		Spatial	
1.75	pre.	mixed	Defl/10% att.	0.74	Spatial	0.08
			Infl/10% att.	0.18	Spatial-temporal	0.91
			Osci/10% att.	0.08	Temporal	0.009
	rel.	mixed	Infl/10% att.	0.29	Spatial-temporal	0.92
			Osci/10% att.	0.71	Spatial	0.08
		pure	Infl/20%att.		Spatial-temporal	
2	pred.	mixed	Infl/10% att.	0.69	Spatial-temporal	0.32
			Infl/20% att.	0.24	Spatial	0.41
			Infl/30% att.	0.07	Temporal	0.27
	rel.	mixed	Infl/10% att.	0.57	Spatial-temporal	0.33
			Infl/20% att.	0.4	Spatial	0.40
			Infl/30% att.	0.03	Temporal	0.27

third, while 2 is last. Furthermore, we also evaluated the regular QRE as we did for the King data set. Similar to the previous data set, we note that with respect to the prediction error (Figure 5.7(a)), players could be almost completely irrational while their best strategies will still follow the Nash equilibrium, as it converges for $\lambda \to 0$. The relative error (Figure 5.7(b)) converges fast to the Nash equilibrium for $\lambda \approx 500$.

Table 5.11: AMP - Equilibrium Points Based on *Game 2*

Threshold	Error metric	Nash equilibrium profile				
		profile	attacker		defender	
			strategy	probability	strategy	probability
1.25	pred.	mixed	Defl/10% att.	0.53	Spatial-temporal	0.64
			Osci/10% att.	0.47	Spatial	0.36
	rel.	mixed	Defl/10% att.	0.5	Spatial-temporal	0.67
			Osci/10% att.	0.5	Spatial	0.33
1.5	pred.	pure	Infl/10% att.		Spatial	
	rel.	pure	Infl/10% att.		Spatial	
1.75	pred.	pure	Infl/10% att.		Spatial-temporal	
	rel.	pure	Infl/10% att.		Spatial-temporal	
2	pred.	pure	Infl/10% att.		Spatial	
	rel.	pure	Infl/10% att.		Spatial	



(a) Prediction error (b) Relative error

Figure 5.7: The Quantal Response Equilibrium evaluation for the AMP data set based on *Game 1*

Table 5.12: King - Equilibrium Points based on *Game 3*

Payoffs	Nash equilibrium strategy profile				
	profile	attacker		defender	
		strategy	probability	strategy	probability
$P_{def} = -Error_{pred}$	mixed	Inflation/30% att.	0.54	c=0.06 & 75th *percentile*	0.93
$P_{att} = Error_{pred}$		Oscillation/30%	0.46	c=0.08 & 75th *percentile*	0.07

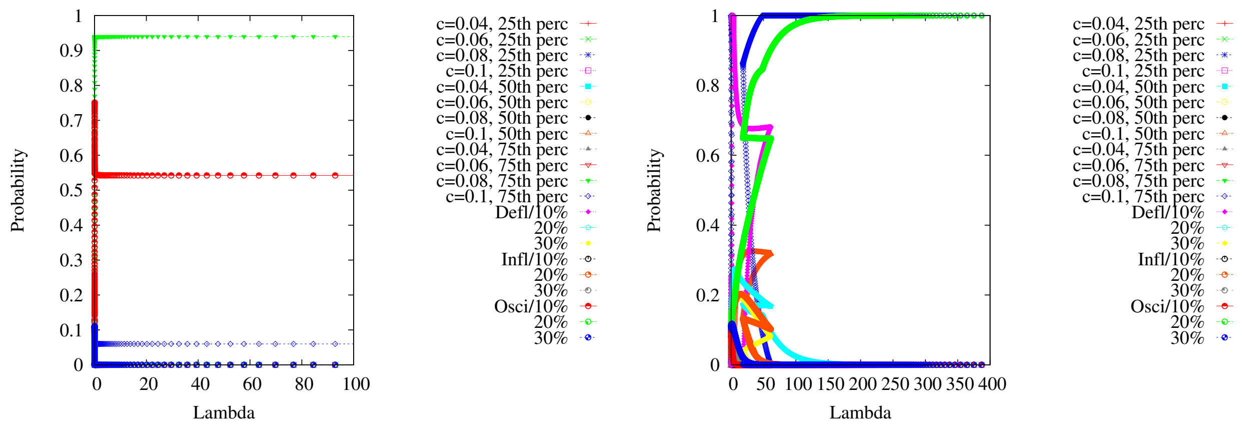
5.6.2 Advanced Game

In the previous analysis, we observed that the best defense mechanism is to apply spatial-temporal outlier detection. We now again consider spatial-temporal outlier detection, however we consider that the spatial outlier detection uses an adaptive threshold. We initialize the spatial threshold with 2. In Table 5.12, we can see the Nash equilibrium using *Game 3* and *Game 4* as defined in Table 5.2. We note that we also compare the strategy $c = 0$, which means that control theory is not used at all. Considering *Game 3*, where we assume the attacker wants to disturb as much as possible the correct functioning of the system with the effect that the prediction error increases significantly, the resulting best attack method is to apply inflation attack with 30% attackers with a probability of 0.54 and the oscillation attack with 30% attackers with a probability of 0.46. For the defender the best way to handle this attack method is to make use of the constant $c = 0.06$ with a probability of 0.93, $c = 0.08$ with 0.07 probability, and the 75^{th} percentile of the prediction error for updating the closed-loop feedback control described in Section 5.5.1.

Table 5.13: King - Equilibrium Points based on *Game 4*

Payoffs	Nash equilibrium strategy profile				
	profile	attacker		defender	
		strategy	probability	strategy	probability
$P_{def} = -Error_{pred}$	pure	Oscillation/10% att.	1	$c = 0.08$ & $75^{th} percentile$	1
	pure	Deflation/10% att.	1	$c = 0.06$ & $75^{th} percentile$	1
$P_{att} = T_{avg}$	mixed	Deflation/10% att. Oscillation/10% att.	0.5 0.5	$c = 0.08$ & $50^{th} percentile$	1
	pure	Oscillation /10% att.	1	$c = 0.08$ & $50^{th} percentile$	1

In *Game 4*, we assume the attacker does not only intend to harm the network as much as possible but that he wants to remain undetected. With the resulting Nash Equilibrium in In Table 5.13 we can see that for the attacker the best choice overall is to have only 10% attackers in the network, as otherwise the attacks become too obvious and are detected by the outlier detection. The overall defense mechanism is to use the 75^{th} percentile with $c = 0.06$ and $c = 0.08$ or to use the 50^{th} percentile with $c = 0.08$. It can be seen that there are 3 different Nash Equilibria for this game model, this means that all of these points lead to the best possible gain for the defender with respect to the attack method applied.



(a) *Game 3* (b) *Game 4*

Figure 5.8: The Quantal Response Equilibrium evaluation for the King data set

We again use the secondary evaluation to determine how irrational the players can act while still maintaining the same optimal equilibrium profiles, as rationality can not be guaranteed. In Figure 5.8, we present the regular QRE. The y-axis represents the probability for a strategy for a given λ. We notice that in *Game 3* (Figure 5.8(a)), the QRE converges to the Nash equilibrium for $\lambda \to 0$, implying that even if the attacker is irrational, he will follow the Nash equilibrium with respect to the prediction error. Regarding *Game 4*, the QRE converges to the Nash equilibrium for $\lambda \approx 100$ (Figure 5.8(b)), which is a fast convergence although λ can become ∞. Using this metric as the basis of the payoff function, an irrational attacker will diverge from the Nash Equilibrium, but as it becomes more rational, it quickly follows the optimal identified strategy.

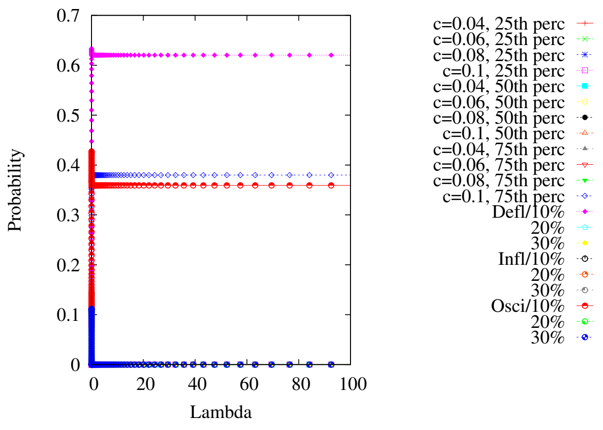
Table 5.14: AMP - Equilibrium Points based on *Game 3*

Payoffs	Nash equilibrium strategy profile				
	profile	attacker		defender	
		strategy	probability	strategy	probability
$P_{def} = -Error_{pred}$ $P_{att} = Error_{pred}$	mixed	Oscillation/30% att.	1	$c = 0.08$ & $75^{th} percentile$ $c = 0.1$ & $75^{th} percentile$	0.25 0.75
	pure	Oscillation/30% att.	1	$c = 0.08$ & $75^{th} percentile$	1
	pure	Inflation/30% att.	1	$c = 0.1$ & $75^{th} percentile$	1

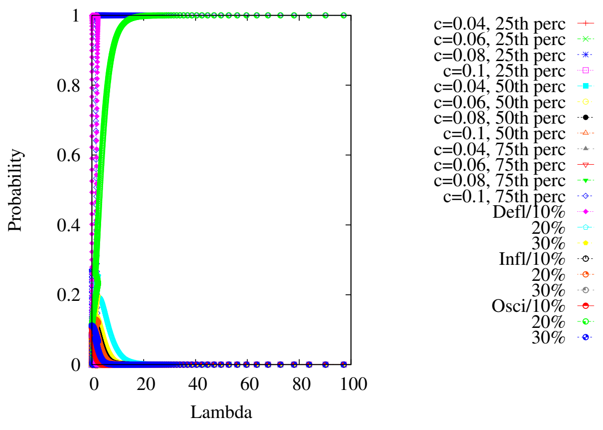
We evaluate the AMP data set with spatial-temporal outlier detection and an adaptive threshold selection for the spatial threshold. We initialize the spatial threshold with 2. Table **??** describes the resulting strategy profiles for the different payoffs and configuration strategies for *Game 3* and *Game 4*. We again note that the different strategies are shown in Table 5.2, including the strategy $c = 0$, where control theory is not used. We notice that the resulting Nash Equilibria are similar to the Nash Equilibria for the KING data set. Based on this, we can assume that independent of the data set we should apply in the closed-loop feedback control a percentile of 75^{th} percentile for the prediction error. An attacker can disturb the network the most while applying the inflation attack, but if he wants the attacks to be undetected then deflation and oscillation are the best attack choices. Furthermore, we again evaluate the regular QRE and notice that with respect to *Game 3* (Figure 5.9(a)) the QRE converges to the Nash equilibrium for $\lambda \to 0$, implying that even if the attacker is irrational he follows the strategy profile defined by the Nash Equilibrium. In *Game 4*, the QRE converges to the Nash equilibrium for $\lambda \approx 20$(Figure 5.9(b)). This converges fast as well, as $0 < \lambda < \infty$.

Table 5.15: AMP - Equilibrium Points based on *Game 4*

Payoffs	Nash equilibrium strategy profile				
	profile	attacker		defender	
		strategy	probability	strategy	probability
$P_{def} = -Error_{pred}$	pure	Oscillation/10% att.	1	$c = 0.06$ & $75^{th} percentile$	1
$P_{att} = T_{avg}$	pure	Deflation/10% att.	1	$c = 0.08$ & $75^{th} percentile$	1
	pure	Oscillation/10% att.	1	$c = 0.08$ & $75^{th} percentile$	1



(a) *Game 3*



(b) *Game 4*

Figure 5.9: The Quantal Response Equilibrium evaluation for the AMP data set

5.7 Conclusions

In this work, we have defined and used an analytical framework in order to analyze strategic choices and identify the best attack strategies and corresponding defense strategies used in virtual coordinate systems. We have modeled the attacker-defender interactions in a game theoretical framework in which the payoffs reflect the incentives and objectives of the two sides. While an attacker aims at inflicting the maximum amount of damage possible by manipulating the virtual coordinate space, a defender attempts to mitigate the damage using countermeasures that filter out rogue information. We have modeled rational attackers in virtual coordinate systems using the Nash equilibrium and irrational attackers using the quantal response equilibrium. From the defender side, we use game theory to see which of the existing mitigation techniques from outlier detection work best against which attacks. We have considered *inflation*, *deflation*, and *oscillation* as attack strategies, and we consider the *spatial*, *temporal*, and *spatio-temporal* outlier detection as mitigation techniques. In this analysis, we have not considered the more subtle attacks (*frog-boiling, network-partition*), as outlier detection is known to be vulnerable to those attacks [22, 23].

We have performed experiments using two Internet topology data sets (King and AMP topology) that have correspondingly different sizes and characteristics. Our results demonstrate that spatial-temporal and spatial outlier detection perform the best while temporal outlier detection is ineffective in isolation. However, temporal outlier detection is often part of the defense profile in combination with the other spatial outlier detection. From an attackers perspective, the best attack strategy to use is the inflation attack with varying percentages of malicious nodes, depending on the deployed defense technique.

We have determined that for large networks (King topology), the inflation attack has the greatest impact on the system. To defend the system, we have found that spatial-temporal outlier detection is the most effective technique given lower spatial outlier thresholds (*e.g.*, ≤ 1.5) and both spatial-temporal and spatial outlier detection provide similar defense performance for higher thresholds. Furthermore, our analysis has found that, independent of the game strategy or the error metric (relative or prediction error) selected, a spatial outlier threshold of 1.25 results in the best system performance, which is smaller than the value found in previous work.

We have found that the resulting strategy profiles for smaller networks (AMP topology) are not as homogeneous as those for the larger King topology, with most of the resulting strategy profiles consisting of a mixed strategy. For example, given the spatial outlier threshold of 1.75, the attacker has the greatest payoff while applying all three attacks with their given probabilities using only 10% malicious nodes. The countermeasure profile looks similar, applying each of the three defense techniques. Both the percentage of malicious nodes necessary to efficiently create the greatest negative impact and the attack and defense profiles have not previously been systematically explored.

Furthermore, we have assessed the most critical component of outlier detection, the usage of a fixed threshold. Such a fixed threshold makes the system vulnerable to subtle attacks that are aware of the outlier detection and are able to remain under radar. We have approached this by proposing an adaptive threshold selection scheme, where we make use of feedback control system. The feedback control system, allows us to adapt to changes in the network, so if we see that the network error is increasing than we tighten the threshold of the outlier detection in order to avoid malicious updates.

We have found that when comparing strategies using a fixed threshold with strategies using an adaptive threshold selection for the outlier detection, the adaptive threshold is more effective in defending against attacks than a fixed threshold. Our analysis shows that when an attacker has as goal disturbing the network as much as possible, using inflation with 30% attackers is the best attack strategy. If the attacker wants to remain also undetected then oscillation and deflation attacks with 10% attackers are the best rational choice. We found that the best parameters for the adaptive threshold is to use the 75^{th} percentile of the prediction error and with a value for the constant c of 0.08 to update the threshold, where c is a system parameter that captures the importance given to the prediction error when updating the threshold.

Chapter 6

Securing Overlay Networks using Machine Learning

In this chapter, we leverage a new detection mechanism for attacks on virtual coordinate systems. The latency estimation in virtual coordinate systems suffers from different attacks as described in Section 4.1. Even though, there have been some defense mechanisms proposed that are able to detect and mitigate those attacks, see Section 4.2, a new kind of attack has been elaborated in [22], which are more subtle and previous defense mechanisms are shown to be vulnerable to them (see Section 4.3). In the following sections, we explain what characteristic makes those attacks immune to the existing defense mechanisms, and we define a new detection model using machine learning techniques that is not vulnerable to those attacks.

6.1 Introduction

We elaborate a detection method using machine learning techniques that is able not only to detect already known attacks (inflation, deflation, oscillation) but also more subtle attacks: the frog-boiling attack and the network-partition attack. Those attacks are able to remain undetected by the previous detection methods, as they use some kind of threshold to define if a coordinate update is malicious or not. The subtle attacks remain under the radar, as they conduct attacks slowly, so that the system is gradually attacked and the change is not detected by threshold based detection mechanisms. The frog-boiling attack is known for abusing the re-learning process of intrusion detection systems, as it will attack the system in such a way that the attacks are not detected and as they are not detected, those attacks are considered as good behavior. So, over time, the intrusion detection system will learn this bad behavior as good behavior.

The challenge in detecting attacks using machine learning techniques lies in identifying an appropriate feature set, that is able to reflect the characteristic of the system in such a way, that even subtle attacks, which are difficult to detect, can be identified. This feature set is used by the machine learning techniques to learn how the system looks like under normal circumstances and how the system looks under malicious updates. We validate this detection method through several different attack scenarios and we compare this dection method to a prior detection mechanism - the outlier detection [111].

The contributions are as follows:

- We propose a practical method to counter the frog-boiling and network-partition attacks, or any complex attack strategy in which several individual attacks are launched by a powerful adversary. For example, the latter can combine several single attacks following a Markov chain model.
- We develop a feature set, based on a node's local information, for embedding it into a multidimensional manifold in order to reveal attacks. This process has resulted in seven feature variables that prove to be the most relevant for the prediction and classification task.
- We provide a quantitative analysis of supervised machine learning methods, *i.e.,* decision trees and support vector machines, for detecting all known attacks in an offline analysis scenario. Our approach works both in a global manner, where all nodes actively exchange local information and a collective decision is taken, as well as in an individual manner, where each node locally decides whether an attack is occurring or not.
- We validate our method by analyzing the performance of online detection. Supervised machine learning techniques like classification are hard or impossible to implement as labelled data is needed. We overcome this challenge by using labelled data to train our classification technique in an offline matter and then implementing the resulted decision process into the real system to allow real-time detection. Specifically, we integrate decision trees into the Vivaldi virtual coordinate system. We analyze the effectiveness of the real-time detection in both a global manner, where a collective decision is taken, as well as in a local manner, where each node decides by itself if an attack is occurring or not. In order to improve the accuracy detection of the local real-time detection we take into account not only a node's local information but also the information already being collected through its interaction with its neighbors.

The remainder of this chapter is structured as follows: We define the new complex attack strategies in Section 6.2. We introduce the reader to machine learning and give an overview of how machine learning is used in intrusion detection in Section 6.3. Our detection framework composed of the feature set is explained in Section sec:concept. Experimental results are shown in Section 6.5 and the results for the system integration are illustrated in Section 6.6. We conclude our work in Section 6.8.

6.2 Attack Strategies

In this section, we explore the different attack strategies that we take into consideration to validate the detection model.

6.2.1 Single Attack Strategies

A single attack describes an attack scenario, where only one attack is applied in one experimental run. All the nodes apply the same attack.

Basic attacks.

A *basic attack* is either an inflation, deflation, or oscillation attack following the description in Section 4.1. inflation and deflation attacks that impact the accuracy of coordinate systems, and oscillation attacks [111] that impact both the accuracy and stability of coordinate systems. In an inflation attack, malicious nodes report a very large coordinate to pull nodes away from correct coordinates. In a deflation attack, to prevent benign nodes from updating and moving towards their correct coordinates, malicious nodes report coordinates near the origin. Finally, in an oscillation attack, malicious nodes report randomly chosen coordinates and increase the RTT by delaying probes for some randomly chosen amount of time. In each of these attacks, nodes report a small, but randomly chosen, local error value, signaling that they have high confidence in their coordinate position.

Advanced attacks

Several proposals have been made to secure virtual coordinate systems against the above described basic attacks [54, 94, 111] and have been shown to effectively mitigate them. However, recent research [22, 23] has identified two more subtle and yet highly effective attacks that are able to bypass existing defenses. They are the frog-boiling and network-partition attacks. In a frog-boiling attack malicious nodes lie about their coordinates or latency by a very small amount to remain undetected by defense mechanisms. The key of the attack is that the malicious nodes gradually increase the amount they are lying about and continue to move further away from their correct coordinates, successfully manipulating benign node's coordinates and thus producing inaccurate RTT estimations. In a network-partition attack two or more groups of malicious nodes conduct a frog-boiling attack, but move their coordinates in opposite directions, effectively splitting the nodes into two or more groups.

6.2.2 Complex Attack Strategies

Prior work has considered only single attack strategies, where a malicious node applies the same attack (inflation, deflation, oscillation, frog-boiling, network-partition) for the entire duration of the attack and all nodes apply the same attack. However, single attack strategies can be easily detected using techniques that leverage change-point detection methods. We extend these scenarios to more complex ones, by assuming that attackers apply sequences of different attacks and not all attacker apply the same attack strategy. Sequences of different attacks do raise the stakes significantly, since the observed patterns are more difficult to detect.

Single Random Attack Scenarios

One way to extend in a straightforward way the single attack strategy is to consider the case where nodes do not perform all the same attack. In this case, each node randomly selects one of the five single attacks, and applies no attack for some time, then switches to the randomly selected attack. For example, in this scenario, some malicious node may conduct the inflation attack, while some other malicious nodes may conduct the frog-boiling attack. We refer to this attack strategy as Single-Random.

Two Attack Scenarios

Another extension of the single attack strategy is a scenario where an attacker alternates between any of the five single attacks, interleaving them with a period of no attack. Specifically, such a strategy

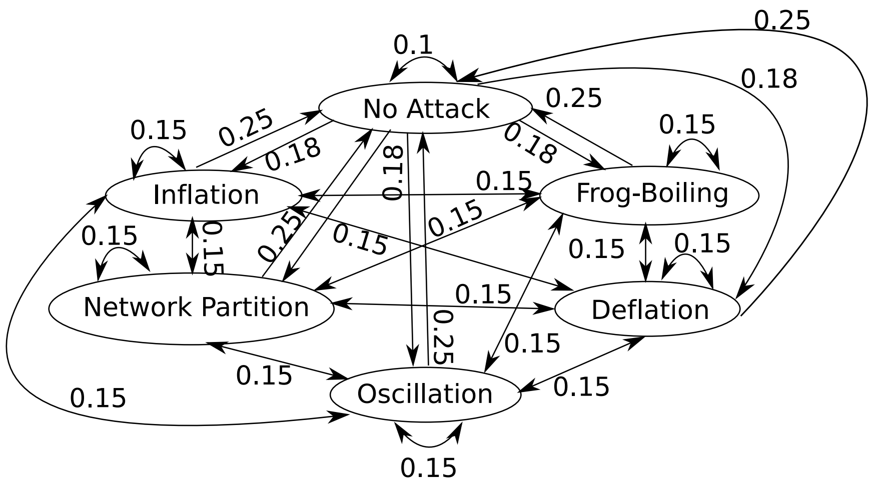


Figure 6.1: Markov Chain with the different attacks and the transition probabilities

is composed of four equal time slots, the first time slot is a non attacking slot, the second consists of one of the five single attacks, followed by another non attacking slot, and finally the fourth time slot is a second single attack. The idea behind this model is to see how the existing detection methods, as well as the methods we propose in this paper, perform in comparison to single attack scenarios. We experiment with several of such scenarios and select the following as representative:

- Deflation - Frog-Boiling
- Oscillation - Inflation
- Network-Partition - Oscillation

Sequence Attack Scenarios

We model more complex attack scenarios, where the attacker applies different sequences of attacks, by using a Markov chain model. The states of such a chain represent all the different single attacks including the *No Attack* state in which an attacker does not apply an attack. The Markov chain is presented in Figure 6.1. This Markov chain is irreducible, as the state space is one single communicating class, meaning that every state is accessible from every state. We consider an irreducible chain, as we assume that the attacker can change the current attack strategy to any other attack, and even stop attacking for a while. Therefore, an attacker can execute every attack at any time, independently of what he has executed previously. Furthermore, the chain is aperiodic, as a return to a specific state can happen at irregular times. An attack that was already executed previously might be utilized again from time to time. Summarizing, we can say that the Markov chain is ergodic, as it is aperiodic, irreducible and positive recurrent. Such an ergodic chain allows to visit individual states indefinitely often and thus leads to more complex scenarios.

The transition probabilities presented in Figure 6.1 reflect several design goals for generating sequences of attacks. From the *No Attack* state, each attack is equally probable, except the probability that no transition (and therefore no attack) is only 10%, therefore the transition to any attack state has the probability 18%. We chose these transition probabilities to avoid the risk of the Markov chain remaining in the *No Attack* state. From an attack state the transitions to every other attack state are equally probable with 15%. This results in the transition probability to the *No Attack* state to always be 25% such that we ensure that there are some no attack intervals and that an attacker does not remain in an attacking state.

Based on the Markov chain presented in Figure 6.1 we created and assessed twenty different sequence-scenarios. All sequences start in the *No Attack* state. Below we describe the most relevant scenarios in terms of representing the different groups of sequences, one group that has a very small amount of non-attacking intervals, another group with intermediate values of non-attacking intervals, and the last group that has the highest amount of non-attacking intervals. We base our selection on the amount of non-attacking intervals as characteristic due to the importance of these intervals for the detection method leveraged in this work. We measure intervals in iterations where an iteration is equivalent to 0.5% of the duration of an experiment. We assume all sequences have 200 iterations. We focus on the following scenarios:

- *Sequence A*: No attack 15 iterations; inflation 15 iterations; network-partition 55 iterations; deflation 35 iterations; inflation 45 iterations; inflation 35 iterations. *Total amount of non-attacking iterations: 15*
- *Sequence B*: No attack 10 iterations; inflation 55 iterations; oscillation 50 iterations; frog-boiling 55 iterations; network-partition 30 iterations. *Total amount of non-attacking iterations: 10*
- *Sequence C*: No attack 30 iterations; network-partition 35 iterations; frog-boiling 35 iterations; No attack 15 iterations; frog-boiling 40 iterations; inflation 45 iterations. *Total amount of non-attacking iterations: 45*
- *Sequence D*: No attack 40 iterations; inflation 30 iterations; oscillation 40 iterations; network-partition 40 iterations; frog-boiling 35 iterations; No attack 15 iterations. *Total amount of non-attacking iterations: 55*
- *Sequence E*: No attack 50 iterations; inflation 10 iterations; No attack 50 iterations; oscillation 55 iterations; oscillation 10 iterations; inflation 25 iterations. *Total amount of non-attacking iterations: 100*
- *Sequence F*: No attack 55 iterations; network-partition 40 iterations; No attack 15 iterations; frog-boiling 45 iterations; inflation 15 iterations; No attack 25 iterations; No attack 5 iterations. *Total amount of non-attacking iterations: 100*

We note that in these sequences of attacks, we still consider malicious nodes that work together by applying the same attacks in the same time interval.

6.3 Background - Machine Learning

Machine learning [2, 50] is a branch of artificial intelligence and is used for machines that should learn from experience. A machine is fed with data and should recognize patterns and learn rules about the data. Machine learning can be divided in two different approaches, the supervised learning and unsupervised learning. Supervised learning (e.g. classification) takes a training set as input and tries to find rules or algorithms that enable the mapping of the input data into the desired output data. The output data (the different classes a datapoint belongs to) is known by a supervisor and the input data is labelled with this, so that the classifier knows how to map. After the training phase, where the classifier/algorithm learns the mapping process, comes a testing phase, where the accuracy of the mapping process is tested. Unsupervised learning (e.g clustering) takes the input data and tries to find correlations between data points, the belonging of the data points is not known.

Machine learning techniques, such as classification, have the aim to separate a given data set into different classes. In our case, the classes that exist are *normal* and *attack*, meaning that we have two different types of data in our data set. On one side, we have data that represents normal updates of the nodes, and on the other side, we have data that represents malicious update requests.

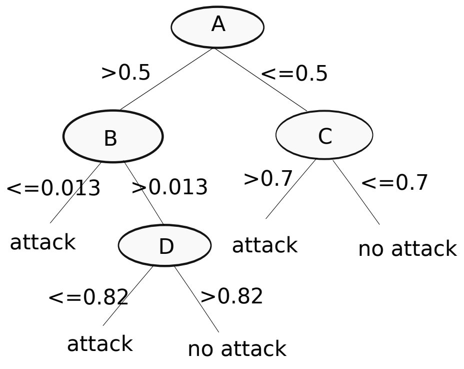


Figure 6.2: Example of a decision tree.

We choose to apply supervised classification methods as we know how the system works under normal circumstances and also how the performance of the system degrades when attacks are taking place. So we can easily generate labelled input data. These classification methods are fed with training data to learn the difference between normal and malicious data. Supervised classification methods can operate directly in the feature space/predictor variables and identify separable regions that can be associated to a given class/dependent categorical variable. Such methods are implemented by decision trees that come in several variants. Simple versions such as Classification and Regression (Cart) [19] can predict both categorical and numerical outcomes, while other schemes (C4.5 for instance) relying on information theory [83] are uniquely adapted to categorical outputs. An example of a decision tree is illustrated in Figure 6.3.

Another type of classification method, support vector machines (SVM) [102], map the input space into another dimensional space, the feature space, and then rely on kernel functions for performing classification in the target space [20]. Figure 6.3 illustrates the mapping of the input data into another dimensional space.

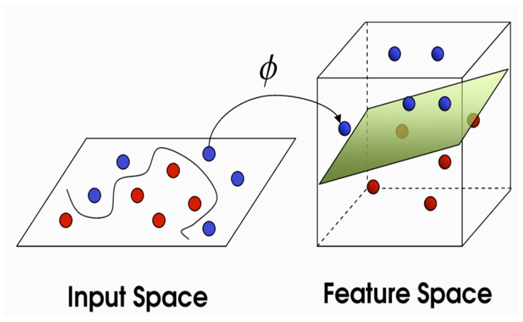


Figure 6.3: Illustration of mapping the input space into another dimensional space for SVM[1].

[1]Source: http://www.imtech.res.in/

6.4 Detection Framework

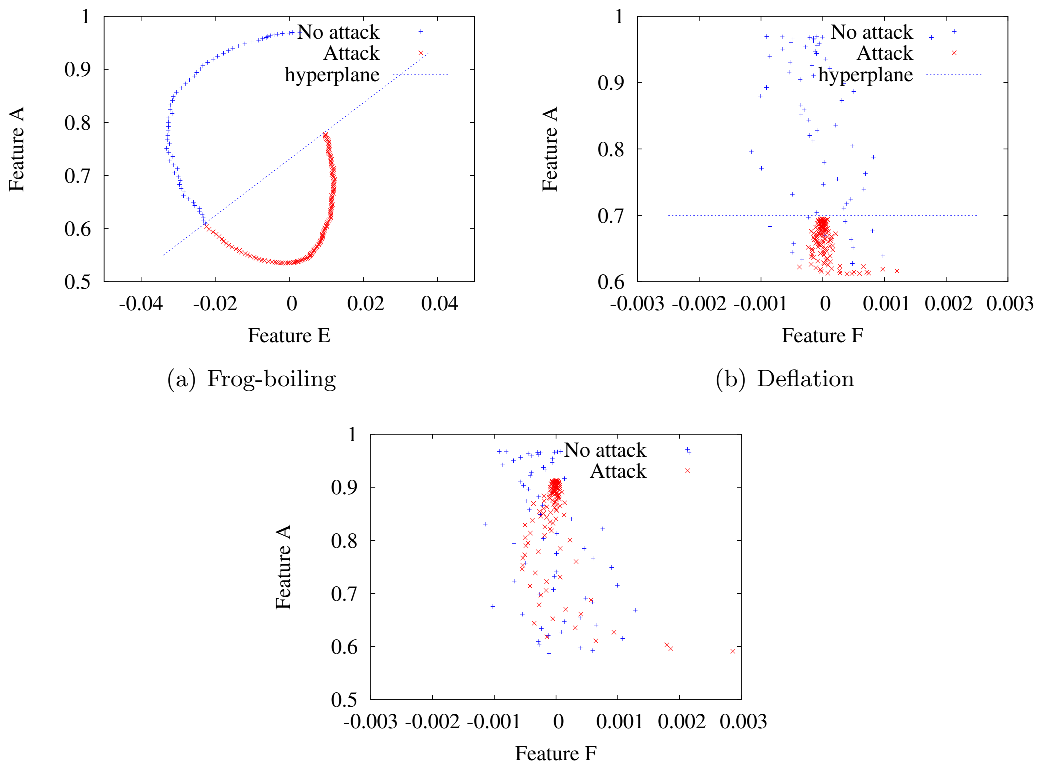
In this section we describe our new mitigation framework based on machine learning techniques described in Section 6.3. The most important thing when applying machine learning techniques is to define an appropriate feature set. We define the feature set that can detect all the attack strategies (described in Section 6.2) in the following section.

6.4.1 Feature Set

We have evaluated three different methods (SimpleCart, C4.5, and support vector machines) for their effectiveness in protecting virtual coordinate systems. We selected these methods since they have performed well in the past in different contexts and by comparing them we can provide a more comprehensive analysis of the detection process. Furthermore, we can also highlight some of the peculiar properties related to the different methods.

We have identified seven feature variables to be used in the prediction task. This process was challenging since several approaches applied directly on the raw data were not successful in detecting attacks. The raw data consisted, in our case, of statistical properties of the underlying local error values. We have analyzed the time series values of both the median and the average local error, but a straightforward analysis of simple time series values did not perform well. This was due to a four lag autocorrelation in the observed time series. In order to decorrelate the time series values, we applied an embedding of the observed one dimensional data into a seven dimensional manifold. Values in the original time series are given by the median local error described in Section **??**. The embedding into a multidimensional manifold aims at revealing subspaces that can be associated to attack states and non-attack ones, respectively. Thus, at each sample moment in time, we analyze a seven dimensional random vector that consists of the following features:

1. *Feature A* is the median local error of the nodes e_{median}. This feature represents the global evolution of the local error. Intuitively, a low median local error means that most of the nodes have converged to their coordinates and an increasing local error means that the nodes are not converging and that an attack is occurring.
2. *Feature B* represents the difference of the median local error at one lag $\delta_1 = e_{median_t}$- $e_{median_{t-1}}$. This feature captures the sense of the variation in the local error. Positive values indicate an increase in the error, while negative values show continuous decrease in the error. Therefore, positive values indicate that an attack is occurring. This feature can be seen as a discretized first derivate of the observed process. Although discrete time events are used to index the time series, by analogy to the continuous case, we assume that this discretized first derivate captures the sense (increasing/decreasing) of the underlying time series.
3. *Feature C* is $\delta_2 = e_{median_t}$- $e_{median_{t-2}}$. This feature relates current values to previous values at a two lag distance. This allows us to observe changes that accumulate over a short period of time.
4. *Feature D* is $\delta_3 = e_{median_t}$- $e_{median_{t-3}}$, is similar to *feature C*, but works at a three lag distance. Using a three lag dependency enables us to see accumulated changes over an intermediate period of time.
5. *Feature E* is $\delta_4 = e_{median_t}$- $e_{median_{t-4}}$. It captures longer dependence (lag four). Observing a four lag dependency, we are able to observe changes that accumulate over a longer period of



(a) Frog-boiling (b) Deflation (c) Inflation

Figure 6.4: Bi-dimensional and pairwise feature representation for different attacks - 1740 nodes (10% malicious) on p2psim.

time. This way, we enable the detection of gradual changes as they occur during a frog-boiling attack.

6. *Feature F* captures the discretized form of the second order derivate δ_{1_t}- $\delta_{1_{t-1}}$. Basically, this feature can indicate the shape (concave/convex) of the initial time series. We assume a discretized equivalent of the continuous definition. This feature allows us to observe a change of feature B over time.

7. *Feature G* is the absolute value of the discretized form of the second order derivate $\mid \delta_{1_t}$- $\delta_{1_{t-1}} \mid$. This absolute value can provide insights in inflection points (i.e., points, where a switch from convex to concave, or concave to convex is happening). An inflection point is helpful for instance, when there is a change from an attack (e.g. inflation) to no-attack, where we could see a change from concave to convex.

We can not visualize a seven dimensional manifold, but to show the rationale for our approach, we illustrate bi-dimensional pairwise scatter plots of the features for different attacks in Figure 6.4. We find the features by running the p2psim simulator using the King data set which contains 1740 nodes out of which 10% are malicious nodes. Figure 6.4(a) shows the two dimensional scatter plot for a frog-boiling attack. Feature A is used for the x-axis and feature E for y-axis. The two classes (attack and non attack) can be linearly separated in this two dimensional subspace. Figure 6.4(b) shows another 2 dimensional scatter plot, where feature A and feature F are used. This scenario corresponds to a deflation attack. In this scenario, the classes can be also linearly separated, and thus we argue that these features are appropriate for defending against a deflation attack. However, in Figure 6.4(c), the same set of features used during an inflation attack shows very limited detection potential. The fact that feature A and F together can detect whether a deflation attack is occurring or not, but can not detect an inflation attack illustrates the need for our seven-fold feature set. Different combinations of features are able to detect different type of attacks. Therefore, the global set of all seven features can be leveraged to detect the different (frog-boiling, deflation, inflation, oscillation and network-partition) attacks.

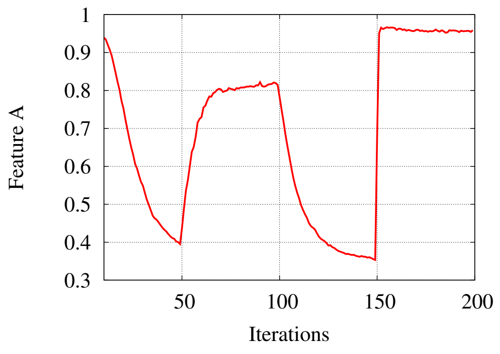
The attack detection problem is stated thus as deciding whether a seven dimensional tuple is representing an attack or not. From a mitigation point of view, once an attack is identified several measures can be taken. In a first phase, updating the virtual coordinates can be resumed after the attack stops, or limited to updates received from known and trusted nodes. The latter assumes an underlying reputation or trust model. In a second phase, the attacking hosts should be identified and contained.

To provide some intuition behind our methodology we present in Figure 6.5 the evolution of two features for a dataset that contains a two attack strategy, Inflation - Oscillation run in the simulator p2psim using the King data set and 10% malicious nodes. This attack scenario consists of four time slots, where the first is a non-attacking slot. The second is in this case an inflation attack. The third time slot is again non attacking, and the fourth and last time slot is the oscillation attack. The objective of classification is, as already mentioned, to separate the different classes of the data set. Two classes exist, the non-attacking and attack class. In Figure 6.5, we illustrate how the classifier can identify different attacks. Figure 6.5(a) shows how feature A, the median of the error, evolves. In this simple case, the increasing or decreasing trends are easy to identify and one can define when the attacking time slots take place. Feature A decreases in a non-attacking time slot, and increases during an attack. However, feature B captures a smoothed version of the overall evolution. In these plots, we can identify intervals that correspond to positive y-values for feature B. These positive values belong to attacking time slots.

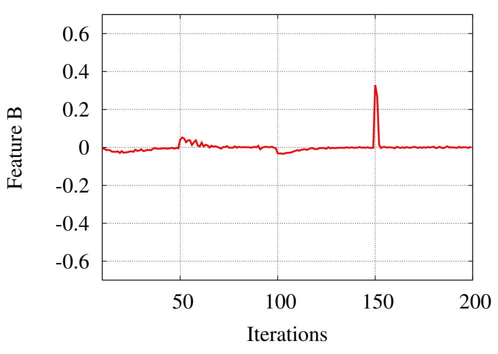
However, feature B, C, D, and E capture a smoothed version of the overall evolution. The difference between those features is, that they have a different correlation to past iterations. The correlation of feature E is more verbose, and so we can see in Figure 6.5(e) that the evolution is more significant than, for instance, in Figure 6.5(a), as the correlation of the latter with the time is smaller. In these plots, we can identify intervals that correspond to positive y-values for all the four graphs corresponding to feature B, C, D, E. These positive values belong to attacking time slots. In Figure 6.5(f) and Figure 6.5(g) represent feature F and G respectively. In these figures we can detect a peak for the instances where the attacking time slots start.

6.5 Experimental Classification Results

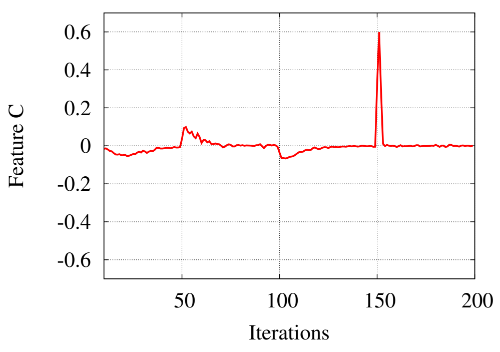
In this section, we evaluate the single and complex attack strategies described in Section 6.2 using Vivaldi within two different environments. First, we evaluate the effectiveness of the machine learning techniques on the dataset resulting from simulation using the p2psim simulator [77] and the King data set topology [45]. Second, we evaluate our machine learning techniques on the data set resulting from deploying Vivaldi on 500 nodes on the Internet PlanetLab testbed [28]. We evaluate our detection method in two setups: global and local. In the global case, every node's information is centrally collected and analyzed together, while in the local case each individual node decides if an attack is taking place or not based only on its own information.



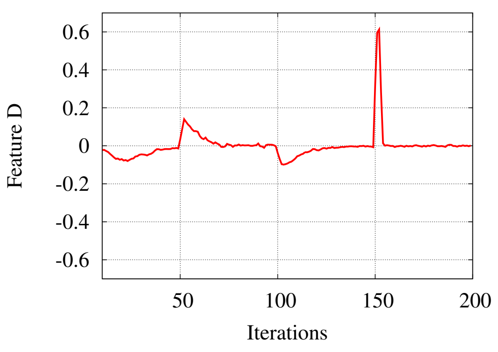
(a) Feature A - e_{median}



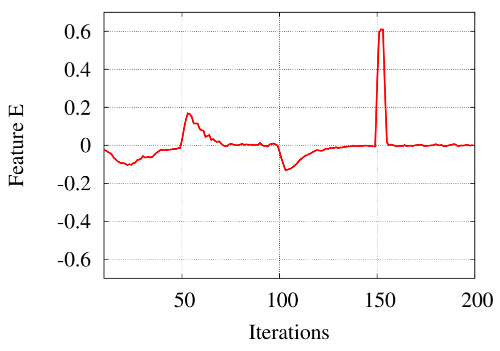
(b) Feature B - δ_1



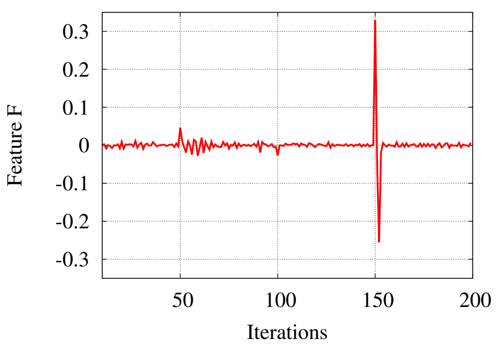
(c) feature C - δ_2



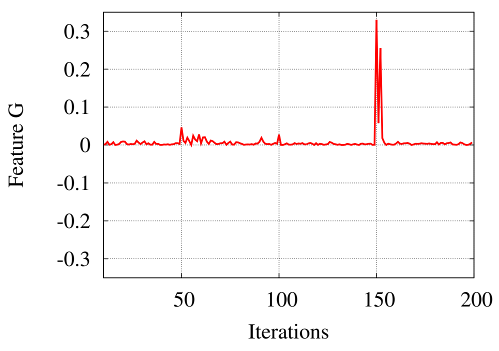
(d) feature D - δ_3



(e) feature E - δ_4



(f) feature F - $\delta_{1\_t} - \delta_{1\_t-1}$



(g) feature G - $| \delta_{1\_t} - \delta_{1\_t-1} |$

Figure 6.5: Features for the Inflation - Oscillation two attack scenario - 1740 nodes (10% malicious) on p2psim.

We use the classification model as described in Section 6.4. To calculate the feature set for the global case we take the median local error of each node in the system, i.e. for 1740 nodes in the simulation and for 500 nodes in the Internet PlanetLab testbed. The acquired model is applied to three different classifiers, namely the two decision trees, SimpleCart [19] and C4.5 [83], and the support vector machines, LibSVM [25]. All experiments for the simulator as well as for PlanetLab are evaluated using the Java source code of weka [47]. We have tried all different kernel functions and their corresponding parameters for the LibSVM and because no significant differences were relevant, we decided to use the default values that come with this weka composant: C-SVC for the kernel type, radial-basis kernel function, with the default values (degree in kernel function was set to 3, gamma parameter to 0.5 and nu to 0.5).

To evaluate the results, we calculate the percentage of attack events that the classifier correctly classifies, which we refer to as the true positive rate (TPR). We also calculate the percentage of non-attack events that the classifier incorrectly classifies as attack events, which we refer to as the false positive rate (FPR). We computed the TPR and FPR using the well established 10-fold cross-validation scheme, where the system is trained with randomly extracted $\frac{9}{10}$ of the data, and tested with $\frac{1}{10}$ of the data. This process is repeated 10 times for each classification.

6.5.1 Simulation Results

We conduct simulations using the King data set topology [45], as it is representative of an Internet-wide deployment of a peer-to-peer system and has been used previously to validate several other VCSes. The King data set consists of RTT measurements between 1740 nodes, of which the average RTT is 180ms. For each simulation, all nodes join in a flash-crowd sequence at the beginning of the simulation. The simulations last for 200 time units, where each time unit is 500 seconds. Each node independently chooses a neighbor set of 64 nodes from which it receives coordinate updates. We compare our machine learning methods to a previously proposed solution using outlier detection [111] that can defend against inflation, deflation, and oscillation.

Single Attack Strategies.

We start by analyzing single attack scenarios, as defined in Section 6.2.1, where the following single attacks are classified: inflation, deflation, oscillation, frog-boiling, network-partition, and single-random. Table 6.1 shows the classification results. The data set consists of the first 30% of the time where no attack occurs, and the remaining 70% the attack does take place. This distribution of time intervals was chosen because some amount of samples of normal data, without attacks, is needed for training.

We note that for the decision trees, SimpleCart and C4.5, the TPR is, for all the different attacks, around 99%, and the FPR for the two classifiers is around 3% indicating that these decision trees can classify correctly almost all attacks. When the number of attackers applying the given attack is increasing, the TPR remains more or less the same, whereas the FPR increases showing that even though most attacks are still correctly classified, normal updates are classified incorrectly more often as the number of attackers increases. We also observe that for all the different types of attacks, independently of the number of attackers, support vector machines perform badly, especially with respect to the FPR.

In order to see to what degree decision trees can detect a frog-boiling attack, we applied a ten-times

Table 6.1: p2psim - 1740 nodes - Single Attack Strategies - Classification Results

Attack Strategy		SimpleCart		C4.5		LibSVM	
		TPR	FPR	TPR	FPR	TPR	FPR
Inflation	10% attackers	0.99	0.01	0.99	0.02	0.67	0.67
	20% attackers	0.99	0.01	0.99	0.02	0.67	0.67
	30% attackers	0.97	0.05	0.99	0.02	0.67	0.67
Deflation	10% attackers	0.99	0.013	0.99	0.02	0.67	0.66
	20% attackers	0.98	0.021	0.98	0.02	0.67	0.67
	30% attackers	0.98	0.016	0.97	0.03	0.67	0.67
Oscillation	10% attackers	0.99	0.008	0.99	0.01	0.67	0.66
	20% attackers	0.98	0.02	0.99	0.03	0.67	0.67
	30% attackers	0.98	0.020	0.98	0.030	0.67	0.67
Frog-Boiling	10% attackers	0.99	0.011	0.99	0.013	0.68	0.64
	20% attackers	0.99	0.016	0.98	0.03	0.67	0.67
	30% attackers	0.98	0.025	0.98	0.03	0.67	0.67
Network-Partition	10% attackers	0.99	0.01	0.98	0.014	0.79	0.44
	20% attackers	0.99	0.01	0.99	0.01	0.67	0.67
	30% attackers	0.99	0.006	0.98	0.03	0.67	0.67
Single-Random	10% attackers	0.99	0.02	0.98	0.03	0.67	0.67
	20% attackers	0.99	0.003	0.98	0.02	0.67	0.67
	30% attackers	0.99	0.002	0.99	0.02	0.67	0.67

Table 6.2: p2psim - 1740 nodes - Complex Scenarios - Two Attacks - Classification Results

Attack Strategy		SimpleCart		C4.5		LibSVM	
		TPR	FPR	TPR	FPR	TPR	FPR
Deflation - Boiling	10% attackers	0.95	0.05	0.94	0.05	0.58	0.41
	20% attackers	0.96	0.05	0.95	0.05	0.51	0.47
	30% attackers	0.98	0.02	0.97	0.03	0.52	0.49
Oscillation - Inflation	10% attackers	0.97	0.04	0.97	0.04	0.51	0.48
	20% attackers	0.97	0.03	0.96	0.04	0.50	0.49
	30% attackers	0.96	0.04	0.97	0.03	0.51	0.49
Network-Partition - Oscillation	10% attackers	0.95	0.05	0.97	0.03	0.67	0.34
	20% attackers	0.91	0.09	0.93	0.07	0.54	0.46
	30% attackers	0.90	0.10	0.92	0.08	0.55	0.45

slower frog-boiling attack as well as hundred-times and thousand times slower and evaluated. In this case decision trees showed a very good performance, for 10%, 20%, and 30% of malicious peers we achieve always a true positive rate around 98% and a false positive rate around 2%.

Complex Attack Scenarios.

We now investigate more complex sequences of attacks, specifically the two attack and sequence attack scenarios as defined in Section 6.2.2. Table 6.2 describes the classification results regarding the two attack scenario. It can be seen that the TPR for both decision trees (i.e., SimpleCart and C4.5) is less than for the single attack scenarios and the FPR is in comparison a bit higher. Overall, the decision trees perform well, although the results are not as good as the single attack scenario. In comparison,

Table 6.3: p2psim - 1740 nodes - Complex Scenarios - Sequence Attacks - Classification Results

	SimpleCart		C4.5		LibSVM	
	TPR	FPR	TPR	FPR	TPR	FPR
A	0.93	0.43	0.94	0.42	0.93	0.86
B	0.96	0.48	0.97	0.33	0.95	0.76
C	0.97	0.08	0.97	0.05	0.79	0.72
D	0.97	0.05	0.98	0.02	0.73	0.73
E	0.98	0.02	0.99	0.02	0.53	0.46
F	0.97	0.04	0.98	0.02	0.7	0.3

the support vector machine library seems to ameliorate, especially in the context of the FPR for the "Network-Partition - Oscillation" attack sequence.

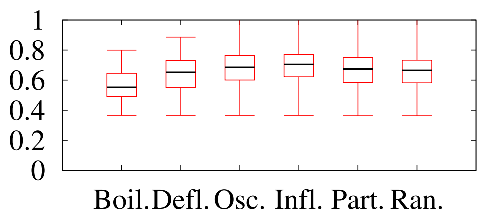
Table 6.3 illustrates the results for the sequence scenarios for 10% malicious nodes. The results show that all techniques have a very good TPR, whereas the FPRs differ significantly. We find that the difference lies in the amount of non-attacking iterations that each sequence has. Sequences A and B are in the group with only a small amount of non-attacking iterations - 10 and 15 iterations. The high FPR thus results due to the classifier not having enough training data for learning normal behavior. The two other groups show better results, for example, as sequences C and D have 45 and 55 normal iterations, respectively. Sequences E and F have in this case a quite high value of non-attacking iterations, both have 100 of them, so exactly half of the data set is non-attacking. We can deduce then that having only 5% non-attacking training data is definitely not enough, whereas 25% already shows good results. This outcome can be explained by the need for an heterogeneous training set for the decision trees; thus, if we have less "No attack" time iterations, it is difficult for the classifier to learn what normal behavior is.

Comparison with Outlier Detection.

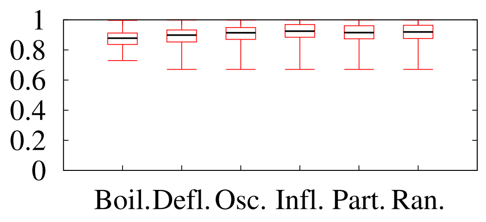
In previous sections we showed that our classification techniques work well when applied globally. Nevertheless, previous works proposed mitigation techniques with respect to single nodes, even if only effective for inflation, deflation, and oscillation attacks. In particular, in the work from [111], each node independently decides if an update should be considered malicious or not by using spatial-temporal outlier detection. We compare our method, applied in a local manner where each node will classify attacks based only on its local information, with the work from [111], referred to as Outlier Detection in the remainder of the section.

As this evaluation depends on the amount of updates those individual nodes receive, we observed some differences in the classification results. For this local classification, we consider that every node will create its own decision tree for the individual decision process. We illustrate the local classification results when there are 10% malicious nodes and for fifty randomly chosen benign nodes since this allows us to have a statistical overview over the whole data set. Based on these fifty nodes we create box-and-whisker diagrams, as those show the median values, the 25^{th} and 75^{th} percentiles, and the minimal and maximal value of each data set. These diagrams are shown in Figure 6.6 and in Figure 6.7. We show results only for the C4.5 technique as it has a similar performance with SimpleCart, while being more relevant in recent research, and it outperforms LibSVM.

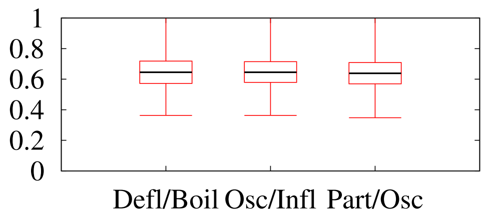
Figure 6.6 shows that for all the different cases of attack strategies considered, the classification



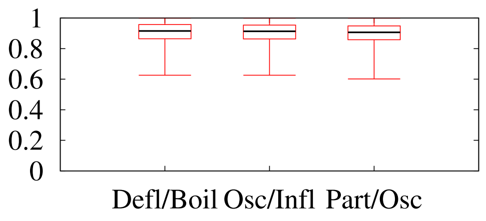
(a) OD - Single attacks



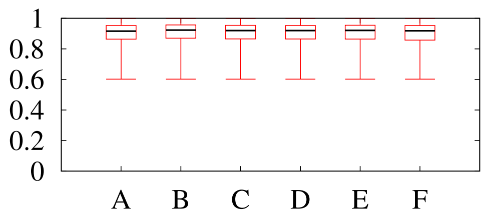
(b) C4.5 - Single-attacks



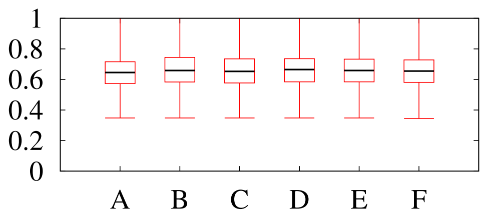
(c) OD - Two attacks



(d) C4.5 - Two Attacks

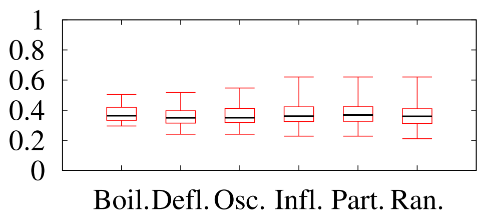


(e) C4.5 - Sequences

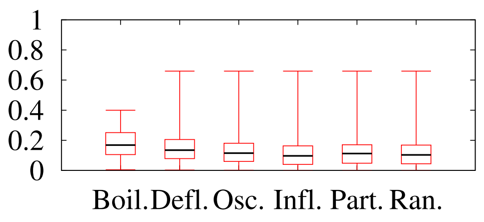


(f) OD - Sequences

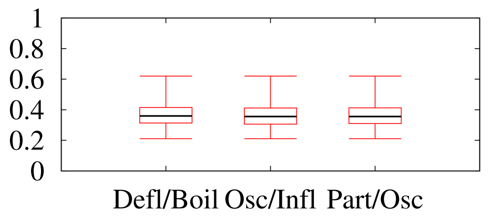
Figure 6.6: p2psim - Outlier Detection Comparison (OD) -TPR - 1740 nodes (10% attackers)



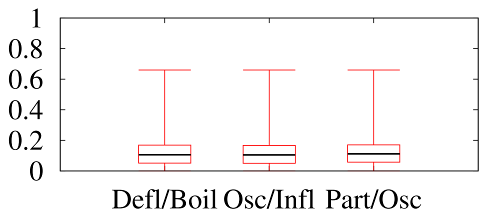
(a) OD - Single attacks



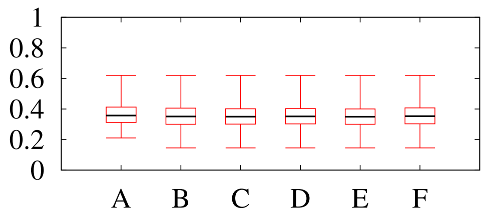
(b) C4.5 - Single-attacks



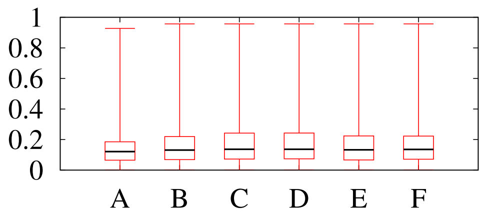
(c) OD - Two attacks



(d) C4.5 - Two Attacks



(e) OD - Sequences



(f) C4.5 - Sequences

Figure 6.7: p2psim - Outlier Detection Comparison (OD) -FPR - 1740 nodes (10% attackers)

Table 6.4: PlanetLab - 500 nodes - Single Attack Strategies - Classification Results

Attack Strategy		SimpleCart		C4.5		LibSVM	
		TPR	FPR	TPR	FPR	TPR	FPR
Inflation	10% attackers	0.97	0.04	0.97	0.03	0.90	0.20
	20% attackers	0.95	0.08	0.95	0.08	0.91	0.17
	30% attackers	0.97	0.05	0.99	0.01	0.93	0.14
Deflation	10% attackers	0.99	0.02	0.98	0.2	0.90	0.21
	20% attackers	0.96	0.05	0.95	0.07	0.92	0.16
	30% attackers	0.97	0.05	0.98	0.03	0.93	0.13
Oscillation	10% attackers	0.99	0.02	0.99	0.01	0.95	0.10
	20% attackers	0.99	0.02	0.99	0.02	0.95	0.11
	30% attackers	0.99	0.02	0.99	0.02	0.95	0.09
Frog-Boiling	10% attackers	0.96	0.05	0.97	0.04	0.80	0.21
	20% attackers	0.97	0.04	0.98	0.03	0.85	0.15
	30% attackers	0.97	0.05	0.98	0.04	0.86	0.15
Network-Partition	10% attackers	0.93	0.10	0.93	0.07	0.83	0.17
	20% attackers	0.96	0.04	0.97	0.03	0.79	0.21
	30% attackers	0.96	0.05	0.94	0.08	0.85	0.14
Single-Random	10% attackers	0.99	0.03	0.98	0.03	0.92	0.16
	20% attackers	0.99	0.01	0.99	0.01	0.96	0.07
	30% attackers	0.99	0.014	0.99	0.013	0.94	0.11

technique performs better than Outlier Detection. In Figure 6.6(a), we see that Outlier Detection performs best for the inflation attack, and we see that frog-boiling has worse results. This is due to the fact that Outlier Detection can not handle frog-boiling as explained earlier and shown in [22, 23]. Regarding Figure 6.7 we note that for all the different attack strategies, our classification technique has much better median FPRs than the Outlier Detection.

6.5.2 PlanetLab Results

To validate our findings over the real Internet, we implemented Vivaldi and deployed it on PlanetLab. For our experiments we used 500 nodes, chosen from all over the world, from which the average RTT is 164ms. Each experiment was run for 30 minutes. To create the sequence attack scenarios we divide those 30 minutes by 200 (the length in iterations of a sequence), so a single iteration consists of 9 seconds (e.g. if an attack is conducted for 10 iterations, then that translates into 90 seconds). All other settings were the same as in the simulations. To find the effectiveness of our techniques, we conduct in our PlanetLab experiments the same attacks and sequences as in the simulations on p2psim.

Single Attack Strategies.

Table 6.4 illustrates the results for the single attack scenarios. We note that both decision trees have very good TPR and FPR, similar to the simulation results. However, in the PlanetLab testbed we obtain much better results when applying the support vector machines. This difference occurs due to the decreased amount of nodes compared to the simulation, which leads to the effect that less variances needed to be considered for rendering the feature set linear separable.

Table 6.5: PlanetLab - 500 nodes - Complex Scenarios - Two Attacks - Classification Results

Attack Strategy		SimpleCart		C4.5		LibSVM	
		TPR	FPR	TPR	FPR	TPR	FPR
Deflation - Boiling	10% attackers	0.93	0.07	0.94	0.06	0.83	0.16
	20% attackers	0.88	0.11	0.89	0.10	0.86	0.13
	30% attackers	0.93	0.07	0.91	0.08	0.87	0.13
Oscillation - Inflation	10% attackers	0.95	0.05	0.95	0.05	0.92	0.085
	20% attackers	0.97	0.03	0.96	0.04	0.90	0.095
	30% attackers	0.97	0.03	0.97	0.03	0.89	0.11
Network-Partition - Oscillation	10% attackers	0.92	0.078	0.915	0.085	0.80	0.20
	20% attackers	0.89	0.11	0.90	0.09	0.84	0.16
	30% attackers	0.91	0.09	0.93	0.07	0.85	0.15

Table 6.6: PlanetLab - 500 nodes - Complex Scenarios - Sequence Attacks - 10% attackers - Classification Results

	SimpleCart		C4.5		LibSVM	
	TPR	FPR	TPR	FPR	TPR	FPR
A	0.93	0.83	0.93	0.65	0.93	0.93
B	0.95	0.95	0.94	0.52	0.95	0.95
C	0.86	0.26	0.84	0.31	0.78	0.78
D	0.87	0.21	0.89	0.22	0.73	0.71
E	0.95	0.06	0.96	0.05	0.95	0.06
F	0.87	0.14	0.88	0.12	0.80	0.19

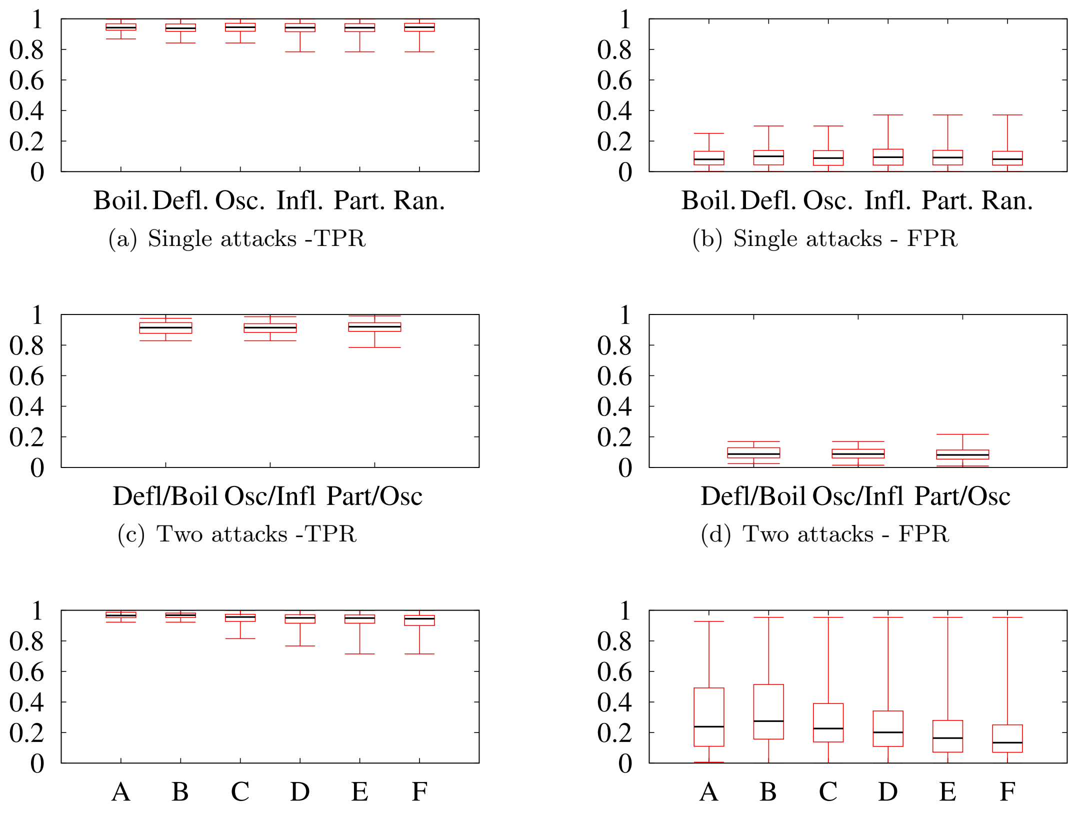
Complex Attack Strategies.

In addition, we also evaluate the more complex sequences. Table 6.5 provides results for the two-attack sequences. We note that, similar to the single attacks, the results for support vector machines are much improved for PlanetLab over the simulator. However, the two decision trees did not perform as well on PlanetLab as they did for the simulations, especially for the 20% and 30% of malicious nodes. This difference occurs as, data produced on PlanetLab has more randomness and noise. Overall, the results are still satisfying though, as the TPR is around 90% and the FPR does not exceed 11%. Table 6.6 illustrates the classification results for the sequence attack strategies for 10% attacking nodes.

Local Classification.

Furthermore, we also analyze PlanetLab results when each individual node decides locally if an attack is taking place or not based only on its individual information. Therfore, a node will creats its own decision tree for the decision process. We show the results in Figure 6.8. We illustrate the C4.5 classification technique, as it outperforms LibSVM, has similar performance to SimpleCart, and has been widely adopted. Similar to the simulations, we evaluate the results when there are 10% malicious nodes and for a set of fifty randomly chosen nodes to have again a statistical overview of the data. To illustrate the evaluation we again use box-and-whisker-diagrams.

Figure 6.8 illustrates that C4.5 has a very high TPR in all the different attack strategies, which mirrors the results for the global classification. We also see that sequences A and B have high FPRs,



(a) Single attacks -TPR
(b) Single attacks - FPR
(c) Two attacks -TPR
(d) Two attacks - FPR
(e) Sequences - TPR
(f) Sequences - FPR

Figure 6.8: PlanetLab Local Results - 500 nodes (10% attackers)

which is similar to the global classification. Overall, except for sequences A and B, the results have good median FPRs indicating that the defined classification technique also work on a local basis when applied on a real Internet testbed.

6.6 System Integration

Previously, we evaluated our solution when using machine learning as an offline analyzing tool. In this section, we show how the proposed techniques can be integrated in a system and used as a real time detection tool. This is an important task, as real-time detection is essential for securing networks. However, there is a lack of anomaly detection that is integrating a supervised decision process into a real system. For supervised classification one needs to have labelled data which is not possible in an online manner. In order to overcome this challenge, we take the labelled data and create the decision making process offline and later we implement this decision process in the real system. We focus on decision trees as we observed that they obtained the best results in the offline analysis. Between C4.5 and CART we choose to use the C4.5 algorithm, as this is a more relevant method nowadays, and the results of C4.5 and CART were similar. We present both simulation (p2p simulator) and Internet (Planetlab testbed) results for the real time detection of attacks.

6.6.1 Simulation Results

Similar to the offline case, we use the local and global approaches for the real system integration, as p2psim simulator supports experimenting with both approaches.

While in the offline classification presented in Section 6.5, we created one attack tree for each attack and for each attack scenario (two attack and sequence scenarios), we can not do the same for the real time analysis since we do not know in advance which attack will take place. Instead, we create a general decision tree for all the attacks and implement it in the system in order to obtain real time detection. Below we describe the creation of the decision tree and the real-time detection results for the global and local integration.

Global Integration

We implement a method in the simulator that will traverse the decision tree depending on the features calculated and also create a decision tree. The system will calculate a global feature set at every iteration based on the median local error of all nodes and will decide, if at that iteration, an attack is occuring.

Tree Creation: We use the experimental data of the single attack scenarios gathered for the offline classification to calculate the feature set under all the different attacks. We then use that data to generate one single generic decision tree that will detect all the different attacks.

The resulting decision tree has 14 leafs and 13 decision nodes and is shown in Figure 6.9. A decision node is considered as node that has 2 leafs, so it is a node where a decision is taken in order to choose the correct leaf. In this figure, we see that Features A, B, C, D, and E are integrated in the decision tree. Even though, Features F and G are not integrated in the global decision tree, they are important for the local decision tree. C4.5 bases its decisions on information theory, therefore we can say that the most relevant feature is Feature E as it is on the head of the decision tree and makes the first division of the tree.

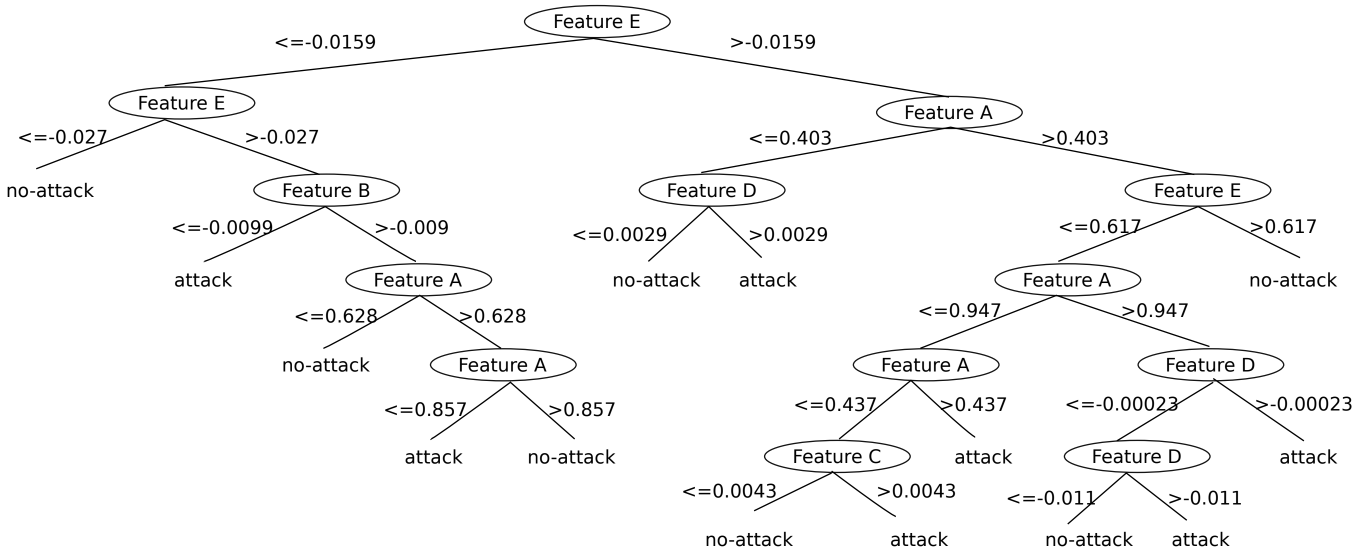


Figure 6.9: Global Decision Tree.

Results: Table 6.7 shows the results for the single attack scenarios. We see that even though, we use only one decision tree for all the different attacks, the results are similar to the offline classification results shown in Table 6.1. System integration results for the two attack scenarios are shown in Table 6.8(a), we note that also for these two attack scenarios we obtain similar results to the offline classification. The results show that although the decision tree was built upon single attack scenarios, it is general enough to detect other, more complex, attack scenarios. Integration results for the sequence attack scenarios for 10% malicious nodes are illustrated in Table 6.8(b). When comparing the integration results with the offline results (Table 6.3) we detect a few differences, we run each of the experiments 5 times, and for every run, we observed similar differences. We saw that the offline results for scenarios C and D are similar, but they differ in the integration results. D has 9% lower TPR than C. This difference results from the different way of creating the decision tree in the offline analysis and the real-time detection. For the offline analysis, for each attack scenario a separate decision tree was built, while here we use one general decision tree for all the different type of attacks.

Local Integration

Global integration has a large network overhead as features are needed from all participating nodes. Global integration may also not work properly if features of part of the network are not available, for instance due to network failures, or if the centralized repository fails. Local integration can address these drawbacks. However, as we showed in the offline classification analysis in Section 6.5, local classification does not perform as well as global classification. Even if the average results were acceptable, there is a large variation (the standard deviation for the FPR is 9% and 7% for the TPR) between nodes, so we had a good median false positive rate, but the maximum of the false positive rates was almost 100% for the sequence attack scenarios. We expect thus that the results for the local integration to have a higher false positive rate than those for global integration.

Table 6.7: p2psim - 1740 nodes - Single Attack Scenarios - Global Integration Results

Attack Strategy		C4.5	
		TPR	FPR
Inflation	10% attackers	0.99	0.04
	20% attackers	0.98	0.04
	30% attackers	0.97	0.04
Deflation	10% attackers	0.99	0.05
	20% attackers	0.98	0.05
	30% attackers	0.99	0.04
Oscillation	10% attackers	0.99	0.05
	20% attackers	0.99	0.05
	30% attackers	0.99	0.05
Frog-Boiling	10% attackers	0.97	0.06
	20% attackers	0.98	0.05
	30% attackers	0.98	0.05
Network-Partition	10% attackers	0.97	0.04
	20% attackers	0.98	0.04
	30% attackers	0.98	0.05
Single-Random	10% attackers	0.99	0.06
	20% attackers	0.99	0.05
	30% attackers	0.99	0.05

Table 6.8: p2psim - 1740 nodes - Global Integration Results

(a) Two Attack Scenarios

Attack Strategy		C4.5	
		TPR	FPR
Deflation - Boiling	10% attackers	0.97	0.03
	20% attackers	0.98	0.03
	30% attackers	0.98	0.02
Oscillation - Inflation	10% attackers	0.99	0.04
	20% attackers	0.96	0.02
	30% attackers	0.96	0.02
Network-Partition - Oscillation	10% attackers	0.96	0.04
	20% attackers	0.96	0.03
	30% attackers	0.97	0.03

(b) Sequence Attack Scenarios - 10% attackers

Attack Strategy	C4.5	
	TPR	FPR
A	0.89	0.15
B	0.85	0.36
C	0.91	0.03
D	0.82	0.04
E	0.99	0.02
F	0.94	0.04

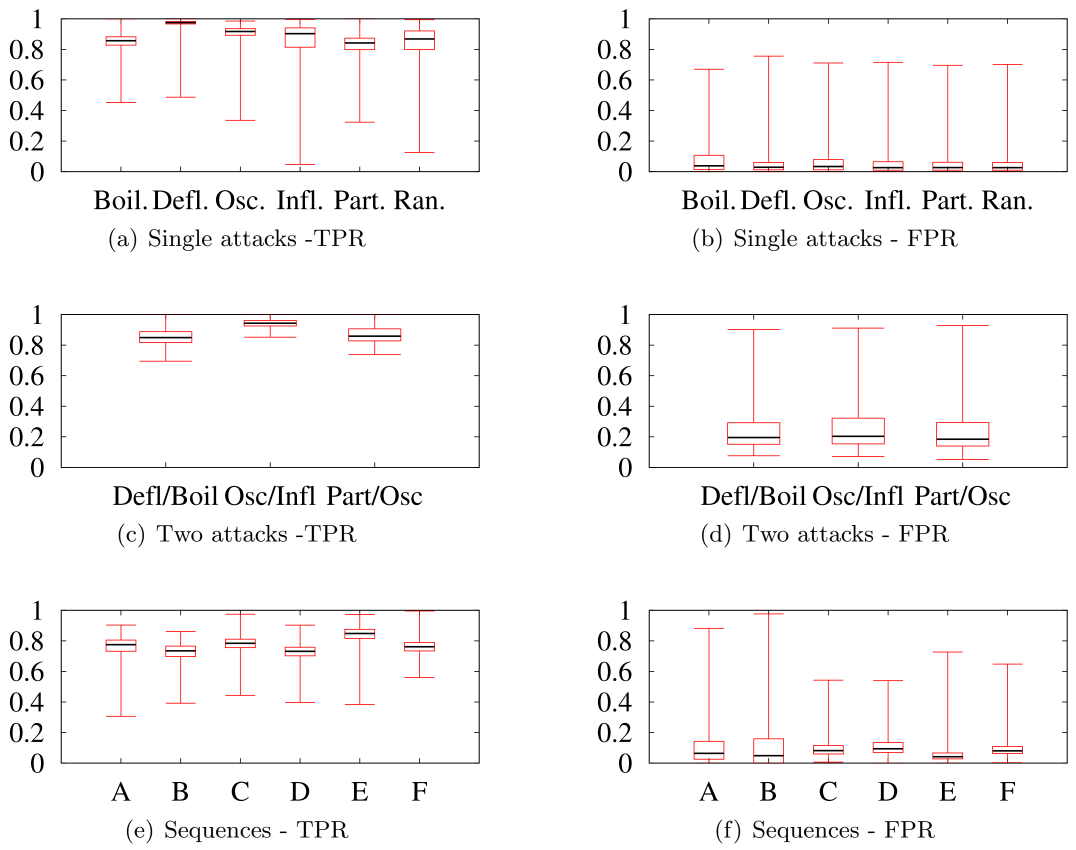
Tree Creation: We first created a single decision tree considering all the nodes simply based on the local error of every single node. This decision tree is used by all the nodes for each individual (per node) decision process. This approach did not perform well and resulted in a large FPR (around 30%) and a low TPR (around 40%). Due to the integration in the real system, there is a large variation of the local error values between the single nodes, so it is not possible to adapt one behavior scheme that can cover all the different variations of the nodes. To address these limitations, we decided to take into account not only the local error value for each single node, but also the median of the 64 neighbor nodes. So we create a decision tree based on the median values of the neighbor set, again this one decision tree is used by all the nodes for the individual decision process. This approach is feasible as each node in Vivaldi, as a normal part of the protocol, has knowledge about the error values of its neighbors. Finding the median value allows a single node to avoid having a large variation in the values compared to other nodes. So creating one decision tree considering the features for all the nodes resulted in a tree composed of 380 decision nodes. Although this sounds like a hugh decision tree and one might think that a node needs to take a hugh number of decisions in order to classify the data into an *attack* or *non-attack*, it is interesting to note that the decision tree is fat and short. So the average path a node takes before classifying consists of 12.64 decision nodes. The longest path consists of 28 decision nodes and the shortest path consists of 3 decisions. As we cannot present such a large decision tree we give an overview of the usage of every feature in Figure 6.11. We see that Features A and E are the two most relevant features. We also see that all the features are used, compared to the decision tree of the global integration, where Features F and G where not relevant. This means that even if the features were not that relevant for creating a global decision tree, they were needed for taking decisions on a local basis. This relates to the fact that Feature E works like a long term memory (lag 4), while Features F and G are based on short term. The global decision tree relies mostly on Feature E and not on F and G because it obtains and uses all the measurements from all nodes and as such is slower to react to brusque changes, since some of the measurements will outweight each other. Local nodes are more reactive in detecting fast changes, but more decision nodes are needed in order to capture all the different changes.

Results: Figure 6.10 shows the results for the local integration in the p2psim simulator of the decision tree based on the median local error value of the 64 neighbor nodes considering 10% malicious nodes. The median TPRs for the single attacks (Figure 6.10(a)) are similar to the global integration results. The rectangle indicates the 25^{th} and 75^{th} percentiles. For all the single attacks the rectangle is close to the median value, which means that most of the nodes have similarly good results for classification. However, outliers do exist, as indicated by the sample minimum. The FPRs for the single attacks (Figure 6.10(b)) also has good values, where the 25^{th} and 75^{th} percentile are very close to the median value. Although, the sample maximum indicates that at least one outlier node for each attack that has a high FPR. The sample minimum of the TPRs for the two attack scenarios (Figure 6.10(c)) show that even in the worst case nodes do have high TPR. On the other hand, the corresponding FPRs show that in the worst case nodes have high FPR (Figure 6.10(d)). Furthermore, the TPRs for the sequence attack scenarios (Figure 6.10(e)) show results similar to those found in the global integration. The same is seen for the FPRs of the sequence attack scenarios (Figure 6.10(f)), where one can clearly see the difference of the amount of non-attacking intervals, where scenarios A and B have only very low amount and therefore have a high FPR and also high outliers.

6.6.2 PlanetLab Result

Similar to the offline analysis, we only present local integration results for PlanetLab.

Tree Creation: We create the decision tree for PlanetLab in a way similar to that done for the p2psim simulator. Each individual node calculates its feature set by finding median error value of its 64 neighbors. A single decision tree is generated that is used by all nodes for the individual decision process. Considering only the local integration is preferable as the goal of Vivaldi is to compute coordinates in a decentralized fashion. The resulting decision tree is composed of 76 decision nodes (average path before classifying: 8.5 decisions; largest path: 13 decisions; shortest path: 3 decisions), this is a large difference to the decision tree for the local p2psim integration (380 nodes). This means therefore, that on PlanetLab, the different attacks are easier to differentiate than in p2psim. We attribute the difference to the fact that in PlanetLab we use 500 nodes, whereas in p2psim we use



(a) Single attacks -TPR

(b) Single attacks - FPR

(c) Two attacks -TPR

(d) Two attacks - FPR

(e) Sequences - TPR

(f) Sequences - FPR

Figure 6.10: Integration - p2psim Local Results - 1740 nodes (10% attackers)

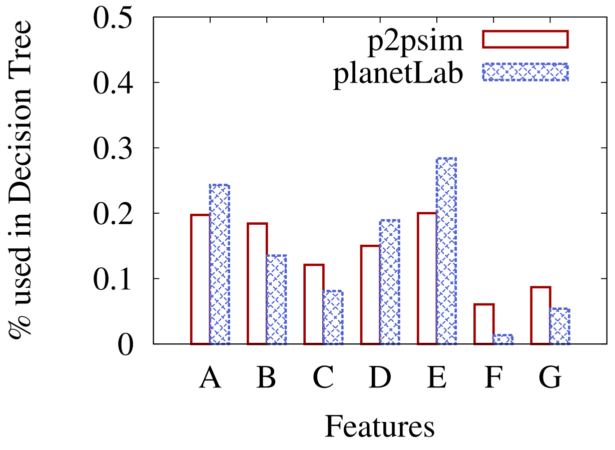
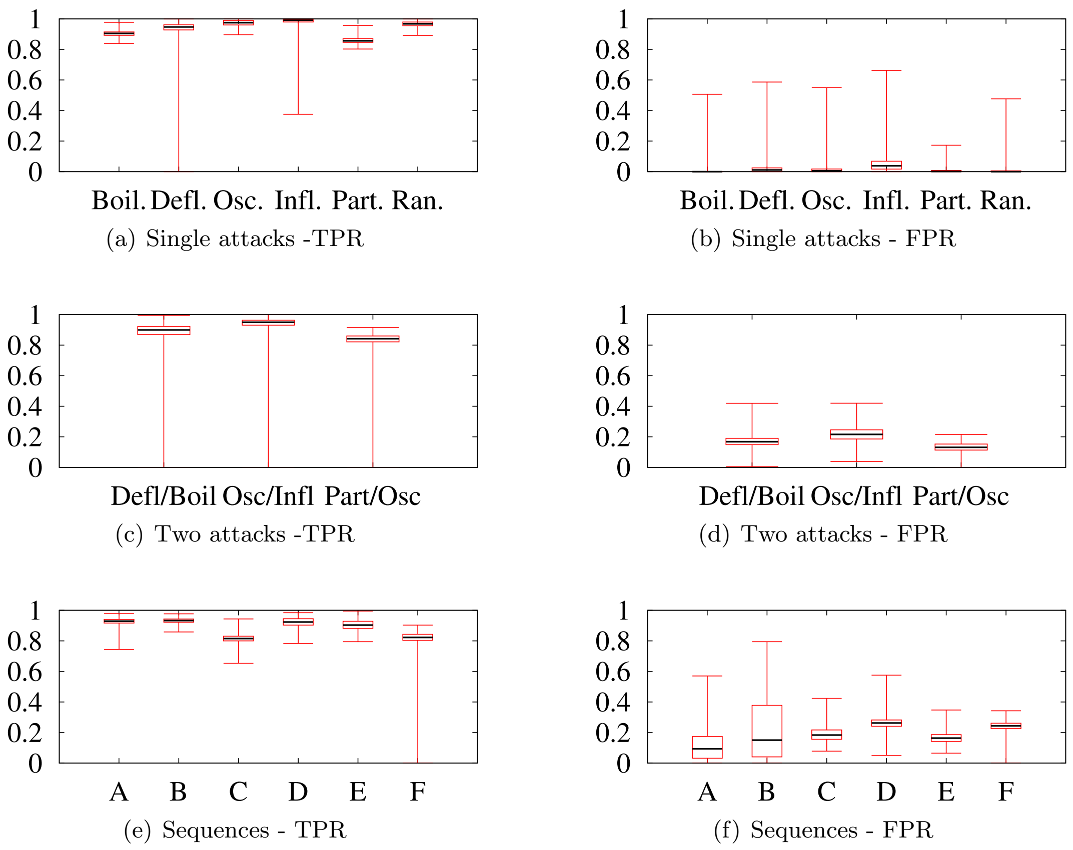


Figure 6.11: Histogram of the features in the two local decision trees.

the King dataset which has 1740 nodes. Having less nodes to cover results in smaller variance and thus a simpler decision process. Figure 6.11 gives an overview of the relevance of every feature in the decision tree. One can see that the overall distribution of features is similar in the p2psim and PlanetLab decision trees, but feature E is more relevant to PlanetLab and features F and G are less relevant for PlanetLab compared to p2psim.

Results: Figure 6.12 shows the results for the local integration in PlanetLab considering 10% malicious nodes. Overall, the PlanetLab integration results look similar to the p2psim integration results. Although, the TPRs for the single attack scenarios (Figure 6.12(a)) do not have regular outliers and most of the minimum values are very close to the 25^{th} percentile. This leads us to the conclusion that outliers happen only very rarely. Nevertheless, there are more outliers for the FPR of the single attacks (Figure 6.12(b)) but the median FPR is even lower than for the p2psim results. The two attack scenarios do show more outliers in the TPR (Figure 6.12(c)) than the FPR (Figure 6.12(b)). The sequence attack scenarios show only few outliers for the TPR (Figure 6.12(e)) but a few more outliers for the FPR (Figure 6.12(f)).



(a) Single attacks -TPR

(b) Single attacks - FPR

(c) Two attacks -TPR

(d) Two attacks - FPR

(e) Sequences - TPR

(f) Sequences - FPR

Figure 6.12: Integration - PlanetLab Local Results - 500 nodes (10% attackers)

6.7 Discussion

In this work, we applied global classification, where at a central point, all the information of all the nodes comes together and a feature set is built based on the median local error values of all nodes. This is easy to achieve in a simulation based system. However, this approach cannot always be applied as a central authority might not be available as for instance in PlanetLab. Therefore, we were not able to integrate a global decision tree in PlanetLab and a global classification for PlanetLab is only available in the offline analysis in Section 6.5. Furthermore, a central authority for providing a decision process represents a single point of failure and could also become a bottleneck. So overall, a local classification is more preferable and the motivation for us to perform the global classification was to obtain a proof of concept.

Nevertheless, some challenges exist for the local classification. In the offline classification analysis, a decision tree was created for every node, so that every node had its own decision tree. Although, in the system integration a single decision tree was produced so that every node used the same decision tree. In general, the offline approach, considering one decision tree per node, is more desirable. Unfortunately, this approach is infeasible in a real system due to the supervised classification method. For every node, the algorithm would need to be trained and at every node a different decision tree would need to be implemented and still a good performance could not be guaranteed as the behavior might have fluctuations in different runs and the decision tree might not be suitable for variations. Unless, a learning process could be integrated at every node, so that a node could be able to train itself using the supervised classification, this approach is not applicable.

Focussing on a single decision tree for all the nodes, one needs to discuss how that decision tree would be distributed in the system as a central authority providing the decision process is undesirable. Also, it could be resource consuming if every node stores the whole decision tree. So a distribution of parts of the decision process would be needed. Using a reputation based system might lower the risk of having attackers that provide falsified decision processes to a node. Furthermore, one could think of using some kind of byzantine fault tolerant system [21] in a scalable variant [3] to provide a distribution that guarantees safety and liveness.

In the experimental results and the system integration section, we see that using supervised machine learning techniques achieve good results. Unfortunately, the supervised machine learning techniques like classification suffer from inconveniences based on the need for training the algorithm. One inconvenience is that the algorithm would need to be re-trained for adapting it to another system, as the behavior in benign conditions as well as under attack might be different depending on the system. Furthermore, it is not always as easy to get labelled data as in our case, and ground truth can be difficult to get. A way to overcome these issues is to use unsupervised machine learning techniques. Although up to now, the unsupervised techniques do not provide as good accuracy compared to supervised techniques.

Especially in intrusion detection this debate has lead to many research for finding the optimal approaches. In intrusion detection systems the network is monitored to find deviations from normal traffic. Machine learning is used in this case to learn the benign traffic and to being able to identify deviations of that benign traffic. The assumption that the characteristics of malicious traffic differs from the characteristics of benign traffic was introduced in [33]. In order to capture these characteristics [33], and the successor [49] make use of profiles based on statistical components. After that many approaches have been proposed that make use machine learning techniques [1, 17, 31, 44, 51, 61, 67, 79, 107].

One branch of intrusion detection systems is anomaly detection [6, 24, 39, 42, 48, 58]. Anomaly detection has been extensively leveraged in developing intrusion detection systems. For example, Bolzoni et al. [17] showed how to automatically and systematically classify detected attacks for anomaly-based intrusion detection systems. The main idea was to extract information out of an attack and to compute similarities of the information of attack data in order to avoid the manual implementation of heuristics, and later classify it automatically, semi-automatically and accordingly raise alerts. One proposed method used support vector machines [102] and a rule learner algorithm for classification. While support vector machines have proved to be efficient (when tuned properly), in our case, we were surprised to discover that their potential usage was quite limited, despite extensive tuning with the most common kernel functions parameter calibration. This anecdotally confirms Sommer et al.'s [96] findings that Machine Learning techniques have often not been successful in real-world IDS applications due to that a detected anomaly does not immediately imply an attack. One major problem with any detection framework is given by the small drifts that might slowly bias the detection process. Repetitive training [31, 67] might be a general solution for decreasing the ratio of false positives.

6.8 Conclusions

In this work, we have addressed the detection of different types of attacks against virtual coordinate systems, the known attacks, such as inflation, deflation and oscillation, as well as the recently identified frog-boiling and network-partition attacks. Besides these existing attacks, we have elaborated more complex attack strategies, the single-random attack scenario, two attack scenario, and sequence attack scenario. We have proposed, as a detection method, to apply supervised machine learning techniques that leverage decision trees, namely SimpleCart and C4.5, and support vector machines to detect all different attack strategies. We identified a feature set and by representing it in a multidimensional manifold, we revealed attacks as these feature variables are used for the prediction and decision task. To our knowledge, this is the first work that is capable of mitigating all known attacks against virtual coordinate systems.

We have validated our detection method through simulation using the King data set for the p2psim simulator as well as through real deployment on the PlanetLab testbed. The detection method is evaluated in a global manner, where the local information of all nodes are together analyzed, as well as in a local manner, where each node has only the local information to analyze and evaluate if an attack is happening or not. We have shown that in our setting, decision trees outperform support vector machines by achieving a much lower false positive rate. Regarding the two different types of decision trees, the results are similar, thus there is no clear better choice. The outcome for the sequence attack scenarios illustrates that a minimal set of normal data is needed for correctly classifying normal behavior, pointing to at most 25% of the data is needed to do so. Furthermore, we compared the proposed detection technique, the decision tree, to existing detection and mitigation techniques, fixed threshold-based outlier detection. This comparison has confirmed that the decision tree as a detection method outperforms the existing outlier detection not only for the frog-boiling, network-partition, or complex attack strategies but also for the inflation, deflation, and oscillation attacks. The results for simulations using the King data set and for real deployments on PlanetLab both demonstrate that our approach identifies the different attacks with a $\sim 95\%$ true positive rate.

Thereafter, we have validated the usage of decision trees as a detection tool by integration in the real system as an online real-time detection tool. We elaborated an enhanced local manner for the integration as real-time detection. The enhanced local detection does not only include the values from the node itself, but includes also values for calculating the feature set of its whole neighbor set.

Taking into consideration the neighbor set of a node, that node has a better overview and is able to better define if an attack is occurring or not. We have integrated decision trees in a global manner and local manner in the p2psim simulator. Furthermore, we also integrated the real-time detection tool in a local manner on the PlanetLab testbed. The real-time detection validates the performance we achieved during offline classification with a $\sim 95\%$ true positive rate for the global manner, and a $\sim 90\%$ true positive rate for the improved local manner.

Part III

Conclusions and Perspectives

Chapter 7

Conclusions and Perspectives

The contribution of this thesis is twofold. The first contribution consists of a game theoretical framework, where we have systematically studied attack and defense techniques in order to assess strategic interactions. Besides the game theoretical framework we have also assessed the critical component of an outlier detection mechanism: the fixed threshold selection. Using a fixed threshold is inflexible and may be exploited by an adversary to remain undetected. Therefore, we have leveraged control theory and have designed an adaptive threshold technique using a feedback system. The contributions of this part can be summarized as follows:

- We have modeled rational attackers in virtual coordinate systems using the Nash equilibrium and irrational attackers using the quantal response equilibrium. From the defender side, we have used game theory to tune our defensive mechanisms in order to mitigate the attacks.
- Using our framework, we have determined that for large networks (*i.e.*, the King topology), the inflation attack has the greatest impact on the system. To defend the system, we have found that spatial-temporal outlier detection is the most effective technique given lower spatial outlier thresholds (*e.g.*, ≤ 1.5) and both spatial-temporal and spatial outlier detection provide similar defense performance for higher thresholds. Furthermore, our analysis has found that, independent of the game strategy or the error metric selected, a spatial outlier threshold of 1.25 results in the best system performance, which is smaller than the value found in previous work.
- We have found that the resulting strategy profiles for smaller networks (*i.e.*, the AMP topology) are not as homogeneous as those for the larger King topology, with most of the resulting strategy profiles consisting of a mixed strategy. For example, given the spatial outlier threshold of 1.75, the attacker has the greatest payoff while applying all three attacks with their given probabilities using only 10% malicious nodes. The countermeasure profile looks similar, applying each of the three defense techniques. Both the percentage of malicious nodes necessary to efficiently create the greatest negative impact and the attack and defense profiles have not previously been systematically explored.
- We have found that when comparing strategies using a fixed threshold with strategies using an adaptive threshold selection for the outlier detection, the adaptive threshold is more effective in defending against attacks than a fixed threshold. Our analysis has shown that when an attacker has as goal disturbing the network as much as possible, using inflation with 30% attackers is the best attack strategy. If the attacker wants to remain also undetected then oscillation and

deflation attacks with 10% attackers are the best rational choice. We have found that the best parameters for the adaptive threshold is to use the 75th percentile of the prediction error and with a value for the constant c of 0.08 to update the threshold, where c is a system parameter that captures the importance given to the prediction error when updating the threshold.

In that work, we have considered three attacks, *inflation, deflation, oscillation.* We did not consider the more subtle attacks (*frog-boiling, network-partition*) as we have analyzed the outlier detection method, and in recent studies [22, 23] it was shown that this detection method is vulnerable to those subtle attacks.

In the second contribution we have proposed a new detection method for all exisiting attacks (*inflation, deflation, oscillation, frog-boiling, network-partition*) by leveraging supervised machine learning techniques. We have shown that our method is also able to detect more complex attack strategies (*two attack scenarios and sequence attack scenarios*). Those complex attack strategies consider intermixed successive attack phases without any fixed order in the attack sequence. The contributions of this part are summarized as follows:

- We have proposed a practical method to counter the frog-boiling and network-partition attacks, or any complex attack strategy in which several individual attacks are launched by a powerful adversary. For example, the latter can combine several single attacks following a Markov chain model.
- We have developed a feature set, based on a node's local information, for embedding it into a multidimensional manifold in order to reveal attacks. This process has resulted in seven feature variables that have proved to be the most relevant for the prediction and classification task.
- We have provided a quantitative analysis of supervised machine learning methods, *i.e.,* decision trees and support vector machines, for detecting all known attacks in an offline analysis scenario. We have evaluated our techniques using the Vivaldi [32] virtual coordinate system through simulations using the King data set and real deployments on PlanetLab. Among the two different machine learning techniques, decision trees and support vector machines, decision trees are able to mitigate all known attacks, outperforming support vector machines by achieving a much lower false positive rate. Our approach has shown to work both in a global manner, where all nodes actively exchange local information and a collective decision is taken, as well as in an individual manner, where each node locally decides whether an attack is occurring or not. The results for simulations using the King data set and for real deployments on PlanetLab both demonstrate that our approach identifies the different attacks with a $\sim$ 95% true positive rate.
- We have validated our method by analyzing the performance of online detection. We have integrated decision trees into the Vivaldi VCS in order to detect attacks in real-time. We have analyzed the effectiveness of the real-time detection in both a global manner, where a collective decision is taken, as well as in a local manner, where each node decides by itself if an attack is occuring or not. In order to improve the local real-time detection, we have designed a new way for a node to decide if an attack is occurring. This is accomplished by taking into account not only a node's local information but also the information already being collected through its interaction with those in its neighbor set. The real-time detection has validated the performance we have achieved during offline classification with a $\sim$ 95% true positive rate for the global manner, and a $\sim$ 90% true positive rate for the improved local manner.

In future work, we want to adapt different game models with the objective to better represent reality. In Chapter 5, the presented game theoretical framework consists of a static game, a one-shot game, this means that the game is formed by one single interaction between the attacker and defender. This is an easy approach that allows to directly compute the Nash equilibrium [70]. However, there exist other more complex approaches, for instance repeated games [41]. In a repeated game, multiple interactions between the players take place. A repeated game is composed of several sub-games. At each sub-game the Nash equilibrium can be calculated. The advantage of using a repeated game is that it is more realistic as not only the current game is relevant, but also the past is relevant to compute those sub-games. In real life, a system, attacks and defenses are not isolated but everything depends on what happened in the past.

Another approach for finding the best defense mechanisms is to use reinforcement learning [57]. Reinforcement learning models an agent that wants to reach a given goal (i.e. the highest possible reward in long-term). An agent can be in different states and choose different actions, this can be seen as a Markov decision process. The concept is similar to game theory, an agent interacts with the environment and can choose different actions depending on the state and receives a reward. The idea is not to achieve a high reward for one action, but to get a high reward in long-term. An example on how such a problem could be solved can be found in the thesis [105]. Using reinforcement learning can be seen as using a adapted repeated game with one single player and with a more complex reward function and a more complex algorithm.

We have applied supervised machine learning techniques in order to leverage a new detection method against all existing attack strategies in Chapter 6. However, when using supervised machine learning techniques, we need labelled data in order to train our model. This can be impractical if we want to adapt such a detection method for a network where we cannot as easily obtain labelled data. Similarly, integrating supervised machine learning techniques for online detection is difficult to achieve as described in [96]. For our approach, we have trained the decision trees with labelled data offline, and we integrated the decision making process in the implementation. The drawback of this approach is that the acquired decision trees are specific to the implementation. We have seen that even though the same algorithm (Vivaldi) is used, the resulting decision trees are different in size for the simulations (p2psim) and the real deployment (PlanetLab), but they share same characteristics like the distribution of the features. This means that if one would like to adopt this detection method in a different environment the offline training with the labelled data would need to be redone. Therefore, we want to make use of unsupervised machine learning techniques to detecting attacks. However, unsupervised machine learning techniques often lack in accuracy as no labelled data is used. We need to investigate what type of unsupervised machine learning could be useful and easily integrated for an online detection methods that still would give a good accuracy in terms of false and true positive rates.

In our work, we first propose a global detection method, based on the median error value of all nodes in order to see if the concept of our project is working. Thereafter, we also propose a local detection method. For the offline technique this local detection only considers the error values of a single node, and for every single node a new decision tree is built. For the real-time local detection, we create offline only one decision tree based on the median values of every node's neighbor set (64 nodes). Overall, one can say that a global detection method is not applicable in real systems, as then a central authority is needed. A central authority always leads to bottleneck issues and single-point-of-failures. Therefore, a local detection method should always be aspired. A discussion of our local detection method can be found in Section 6.7.

Another concern about our method is that we do not deploy a new mitigation framework, the only mitigation we have considered in Section 6.7 is the same as in outlier detection, when updates are discarded because they seem malicious. The drawback of this method is, that a node could become immobile due to rejecting updates. Furthermore, non-malicious updates might also be rejected and not only malicious updates. Another approach we want to investigate in future, is that a node can change its neighbors when a node realizes that its error is increasing and an attack is occurring. For this investigation, we need to see what is the best way for a node to change the neighbor set, if the node should exchange half of the neighbors or directly the complete neighbor set. We also need to see how a node can be sure that when the neighbor set is changed no malicious node is in this new set. Therefore, a reputation based system could be helpful. Whenever a node realizes that an attack is occurring it can give a bad reputation to its current neighbor set, and choose a neighbor set with a better reputation.

Virtual coordinate systems usually use the round-trip-time (RTT) in order to estimate the latency between nodes, and the attacks on those systems, often consists on faking the RTT. The RTT is also often used in other algorithms or protocols as a metric (*i.e.*, NTP). A further task in the future work is to analyze if similar attacks can have a similar effect on those protocols as they have for virtual coordinate systems and if a similar detection method could be used as a failure predictor. After having a failure predictor for networks like NTP, we also want to investigate a failure predictor for large-scale networks like clouds, where availability and up-time are essential. A failure predictor would be able to alert if a failure and possibly a corresponding down-time is about to happen. Depending on the type of failure, a mitigation framework should be able to rectify the problem that could cause a failure and/or a down-time.

Chapter 8

About the Author

Sheila Becker was born in Luxembourg, Luxembourg, on May 11, 1983. After receiving her Industrial Engineer diploma in Applied Informatics from the University of Luxembourg, specialized in distributed systems and networks, in 2007, she went to the University Henri Poincaré in Nancy, France for her Master's Degree. She graduated in 2008 with a research master degree in computer science, specialized in distributed services and communication networks. In December 2008, Sheila became a Ph.D. candidate at the University of Luxembourg in the Computer Science and Communications Research Unit, specifically as part of the SECAN-LAB. In 2009, Sheila also became, as part of an collaborative agreement, a Ph.D. student at University Henri Poincaré. Since then, she was member of the INRIA -LORIA research laboratory and member of the group MADYNES. Her research interests lie in Machine Learning, Network Security and Fuzz Testing.

During the first two years of her Ph.D. studies she worked on the EU-fundet project EFIPSANS, where she contributed to the research aspects relevant to the security issues. Furthermore, Sheila was able to gather management skills as task leader of the security team. In 2010, a research collaboration was started with the Purdue University in West Lafayette, USA with the objective to find new ways in securing overlay networks. In 2011, Sheila was motivated due to the well-going collaboration with the Purdue University to apply for a FULBRIGHT scholarship as a visiting researcher. Sheila got granted the FULBRIGHT scholarship and was able to spend the academic year 2011/12 at the Purdue University, where she not only learned a lot about the american and other cultures but also refined her research skills.

8.1 Publications

- Jeff Seibert, Sheila Becker, Cristina Nita-Rotaru, and Radu State, **Securing Virtual Coordinates by Enforcing Physical Laws**, International Conference on Distributed Computing Systems (ICDCS), Macau, China, June 2012 [92].

- Jeff Seibert, Sheila Becker, Cristina Nita-Rotaru, and Radu State, **Newton Meets Vivaldi: Using Physical Laws to Secure Virtual Coordinate Systems**, Network and Distributed System Security Symposium (NDSS) (invited abstract), San Diego, CA, USA, February 2012 [91].

- Sheila Becker, Jeff Seibert, Cristina Nita-Rotaru, and Radu State, **Securing Application-**

Level Topology estimation Networks: Facing the Frog-Boiling Attack, International Symposium on Recent Advances on Intrusion Detection (RAID), Menlo Park, CA, USA, September 2011 [11].

- Sheila Becker, Jeff Seibert, David Zage, Cristina Nita-Rotaru, and Radu State, **Applying Game Theory to Analyze Attacks and Defenses in Virtual Coordinate Systems**, International Conference on Dependable Systems and Networks (DSN), Hong Kong, China, June 2011 [12].
- Sheila Becker, Humberto Abdelnur, Jorge L. Obes, Radu State, and Olivier Festor, **Improving Fuzz Testing using Game Theory**, International Conference on Network and System Security (NSS), Melbourne, Australia, September 2010 [9].
- Sheila Becker, Humberto Abdelnur, Radu State, and Thomas Engel, **An Autonomic Testing Framework for IPv6 Configuration Protocols**, Mechanisms for Autonomous Management of Networks and Services (AIMS), Zurich, Switzerland, June 2010 [10].
- Sheila Becker, Radu State, and Thomas Engel. **Using Game Theory to configure P2P SIP**, Principles, Systems and Applications of IP Telecommunications (IPTComm), Atlanta, USA, July 2009 [14].
- Sheila Becker, Radu State, and Thomas Engel. **Defensive configuration with Game Theory**. IEEE/IFIP International Symposium on Integrated Network Management (IM), New York, USA, June 2009 [13].
- **Security Evaluation for P2P Communication Systems**, Master Thesis, Université Henri Poincaré in Nancy, France, June 2008 [8].

Bibliography

[1] Naoki Abe, Bianca Zadrozny, and John Langford. Outlier detection by active learning. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 504–509, New York, NY, USA, 2006. ACM.

[2] Ethem Alpaydin. *Introduction to Machine Learning.* MIT Press, Cambridge, MA, 2004.

[3] Y. Amir, C. Danilov, D. Dolev, J. Kirsch, J. Lane, C. Nita-Rotaru, J. Olsen, and D. Zage. Steward: Scaling byzantine fault-tolerant replication to wide area networks. *Dependable and Secure Computing, IEEE Transactions on*, 7(1):80 –93, 2010.

[4] AMP. Nlanr active measurement project. `http://amp.nlanr.net/`.

[5] Azureus. Vuze (azureus) - bittorent client. `http://azureus.sourceforge.net`.

[6] Paul Barford and David Plonka. Characteristics of network traffic flow anomalies. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW '01, pages 69–73, New York, NY, USA, 2001. ACM.

[7] Salman A. Baset and Henning Schulzrinne. An Analysis of the Skype Peer-to-Peer Internel Telephony Protocol. December 2004.

[8] Sheila Becker. Security Evaluation for P2P Communication Systems. Stage, MADYNES - INRIA Lorraine - LORIA, 2008. INRIA CNRS : UMR7503.

[9] Sheila Becker, Humberto Abdelnur, Jorge Lucangeli Obes, Radu State, and Olivier Festor. Improving fuzz testing using game theory. In *Proceedings of the 2010 Fourth International Conference on Network and System Security*, NSS '10, pages 263–268, Piscataway, NJ, USA, 2010. IEEE Press.

[10] Sheila Becker, Humberto J. Abdelnur, Radu State, and Thomas Engel. An autonomic testing framework for ipv6 configuration protocols. In *4th International Conference on Autonomous Infrastructure, Management and Security*, AIMS'10, pages 65–76. Springer, 2010.

[11] Sheila Becker, Jeff Seibert, Cristina Nita-Rotaru, and Radu State. Securing application-level topology estimation networks: Facing the frog-boiling attack. In *Recent Advances in Intrusion Detection - 14th International Symposium*, RAID'11, pages 201–221. Springer, 2011.

[12] Sheila Becker, Jeff Seibert, David Zage, Cristina Nita-Rotaru, and Radu State. Applying game theory to analyze attacks and defenses in virtual coordinate systems. In *Dependable Systems Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*, pages 133 –144, Washington, DC, USA, june 2011. IEEE Computer Society.

[13] Sheila Becker, Radu State, and Thomas Engel. Defensive configuration with game theory. In *Proceedings of the 11th IFIP/IEEE International Symposium on Integrated Network Management*, IM'09, pages 453–459, Piscataway, NJ, USA, 2009. IEEE Press.

[14] Sheila Becker, Radu State, and Thomas Engel. Using game theory to configure p2p sip. In *Proceedings of the 3rd International Conference on Principles, Systems and Applications of IP Telecommunications*, IPTComm '09, pages 6:1–6:9, New York, NY, USA, 2009. ACM.

[15] Ken Binmore. *Playing for Real: A Text on Game Theory.* Oxford University Press, 2007.

[16] Matt Bishop. *Computer Security: Art and Science.* Addison-Wesley, 2003.

[17] D. Bolzoni, S. Etalle, and P. H. Hartel. Panacea: Automating attack classification for anomaly-based network intrusion detection systems. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID '09, pages 1–20. Springer-Verlag, 2009.

[18] J. Bourgain. On lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52(1):46–52, March 1985.

[19] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth International Group, Belmont, California, 1984.

[20] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, June 1998.

[21] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20:398–461, November 2002.

[22] Eric Chan-Tin, Daniel Feldman, Nicholas Hopper, and Yongdae Kim. The frog-boiling attack: Limitations of anomaly detection for secure network coordinate systems. In *Security and Privacy in Communication Networks - 5th International ICST Conference*, SecureComm'09, pages 448–458. Springer, 2009.

[23] Eric Chan-Tin and Nicholas Hopper. Accurate and provably secure latency estimation with treeple. In *Proceedings of the Network and Distributed System Security Symposium*, NDSS'11, pages 1–11. The Internet Society, 2011.

[24] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[25] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[26] Yang Chen, Xiao Wang, Xiaoxiao Song, Eng Keong Lua, Cong Shi, Xiaohan Zhao, Beixing Deng, and Xing Li. Phoenix: Towards an accurate, practical and decentralized network coordinate system. In *Networking*, volume 5550 of *Lecture Notes in Computer Science*, pages 313–325. Springer, 2009.

[27] Yang Chen, Yongqiang Xiong, Xiaohui Shi, Beixing Deng, and Xing Li. Pharos: A decentralized and hierarchical network coordinate system for internet distance prediction. In *GLOBECOM*, pages 421–426, 2007.

[28] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33:3–12, 2003.

[29] Bram Cohen. Incentives build robustness in BitTorrent. In *Proc. of P2P Economics*, pages 1–5, 2003.

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

[31] Gabriela F. Cretu-Ciocarlie, Angelos Stavrou, Michael E. Locasto, and Salvatore J. Stolfo. Adaptive anomaly detection via self-calibration and dynamic updating. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID '09, pages 41–60, Berlin, Heidelberg, 2009. Springer-Verlag.

[32] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: a decentralized network coordinate system. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '04, pages 15–26, New York, NY, USA, 2004. ACM.

[33] Dorothy E. Denning. An intrusion-detection model. *IEEE Trans. Softw. Eng.*, 13(2):222–232, February 1987.

[34] C. A. Desoer, Ruey-Wen Liu, John Murray, and Richard Saeks. Feedback system design: The fractional representation approach to analysis and synthesis. *Automatic Control, IEEE Transactions on*, 25(3):399 – 412, jun 1980.

[35] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[36] Benoit Donnet, Bamba Gueye, and Mohamed Ali Kaafar. A survey on network coordinates systems, design, and security. *IEEE Communications Surveys and Tutorials*, 12(4):488–503, 2010.

[37] John Comstock Doyle, Bruce A. Francis, and Allen R. Tannenbaum. *Feedback Control Theory.* Prentice Hall Professional Technical Reference, 1991.

[38] L. Fisher. *Rock, Paper, Scissors: Game Theory in Everyday Life.* Basic Books, 2008.

[39] Prahlad Fogla and Wenke Lee. Evading network anomaly detection systems: formal reasoning and practical techniques. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 59–68, New York, NY, USA, 2006. ACM.

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

[41] Drew Fudenberg and Jean Tirole. *Game Theory.* MIT Press, August 1991.

[42] Carrie Gates and Carol Taylor. Challenging the anomaly detection paradigm: a provocative discussion. In *Proceedings of the 2006 workshop on New security paradigms*, NSPW '06, pages 21–29, New York, NY, USA, 2007. ACM.

[43] Jacob Goeree, Charles Holt, and Thomas Palfrey. Regular quantal response equilibrium. *Experimental Economics*, 8(4):347–367, December 2005.

[44] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. Active learning for network intrusion detection. In *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*, AISec '09, pages 47–54, New York, NY, USA, 2009. ACM.

[45] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble. King: estimating latency between arbitrary internet end hosts. *SIGCOMM Comput. Commun. Rev.*, 32(3):11–11, July 2002.

[46] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. Peerreview: practical accountability for distributed systems. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles 2007*, SOSP'07, pages 175–188, New York, NY, USA, 2007. ACM.

[47] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[48] Irfan Ul Haq, Sardar Ali, Hassan Khan, and Syed Ali Khayam. What is the impact of p2p traffic on anomaly detection? In *Proceedings of the 13th international conference on Recent advances in intrusion detection*, RAID'10, pages 1–17, Berlin, Heidelberg, 2010. Springer-Verlag.

[49] Alfoso Valdes Harold S. Javitz. The nides statistical component description and justification. Technical Report A010, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025-3493, March 1994.

[50] Peter Harrington. *Machine Learning in Action*. Manning Publications, Shelter Island, NY, 2012.

[51] Wenjie Hu, Yihua Liao, and V. Rao Vemuri. Robust anomaly detection using support vector machines. In *In Proceedings of the International Conference on Machine Learning*, pages 1–7. Morgan Kaufmann Publishers Inc, 2003.

[52] IPoque. Internet observatory. `http://www.internetobservatory.net`.

[53] I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, October 2002.

[54] Mohamed Ali Kaafar, Laurent Mathy, Chadi Barakat, Kave Salamatian, Thierry Turletti, and Walid Dabbous. Securing internet coordinate embedding systems. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '07, pages 61–72, New York, NY, USA, 2007. ACM.

[55] Mohamed Ali Kaafar, Laurent Mathy, Thierry Turletti, and Walid Dabbous. Real attacks on virtual networks: Vivaldi out of tune. In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, LSAD '06, pages 139–146, New York, NY, USA, 2006. ACM.

[56] Mohamed Ali Kaafar, Laurent Mathy, Thierry Turletti, and Walid Dabbous. Virtual networks under attack: disrupting internet coordinate systems. In *Proceedings of the 2006 ACM CoNEXT conference*, CoNEXT '06, pages 12:1–12:12, New York, NY, USA, 2006. ACM.

[57] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[58] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '04, pages 219–230, New York, NY, USA, 2004. ACM.

[59] Jonathan Ledlie, Paul Gardner, and Margo Seltzer. Network coordinates in the wild. In *Proceedings of the 4th USENIX conference on Networked systems design &#38; implementation*, NSDI'07, pages 22–22, Berkeley, CA, USA, 2007. USENIX Association.

[60] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *In NIPS*, pages 556–562. MIT Press, 2001.

[61] Wenke Lee and Dong Xiang. Information-theoretic measures for anomaly detection. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, SP '01, pages 130–, Washington, DC, USA, 2001. IEEE Computer Society.

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

[63] Li-wei Lehman and Steven Lerman. A decentralized network coordinate system for robust internet distance. In *Proceedings of the Third International Conference on Information Technology: New Generations*, ITNG '06, pages 631–637, Washington, DC, USA, 2006. IEEE Computer Society.

[64] Hyuk Lim, Jennifer C. Hou, and Chong-Ho Choi. Constructing internet coordinate system based on delay measurement. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, IMC '03, pages 129–142, New York, NY, USA, 2003. ACM.

[65] Cristian Lumezanu and Neil Spring. Playing vivaldi in hyperbolic space. In *UMD-CS-TR-4843, University of Maryland.*, pages 1–12, 2006.

[66] Kong-Wei Lye and Jeannette Wing. Game strategies in network security. *International Journal of Information Security*, 4(1):71–86, 2005.

[67] Federico Maggi, William Robertson, Christopher Kruegel, and Giovanni Vigna. Protecting a moving target: Addressing web application concept drift. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID '09, pages 21–40, Berlin, Heidelberg, 2009. Springer-Verlag.

[68] Yun Mao, Lawrence K. Saul, and Jonathan M. Smith. Ides: An internet distance estimation service for large networks. *IEEE Journal on Selected Areas in Communications*, 24(12):2273–2284, 2006.

[69] J. McInerney, S. Tubberud, S. Anwar, and S. Hamilton. Friars: a feedback control system for information assurance using a markov decision process. In *Security Technology, 2001 IEEE 35th International Carnahan Conference on*, pages 223 –228. IEEE, oct 2001.

[70] John Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

[71] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, January 1965.

[72] Arbor Networks. Worldwide infrastructure security report 2010 volume vi. http://www.arbornetworks.com/.

[73] Arbor Networks. Worldwide infrastructure security report 2011 volume vii. http://www.arbornetworks.com/.

[74] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[75] T. S. Eugene Ng and Hui Zhang. Predicting internet network distance with coordinates-based approaches. In *Proceedings of the IEEE International Conference on Computer Communications - INFOCOM*, pages 1–10. IEEE Computer Society, 2002.

[76] T. S. Eugene Ng and Hui Zhang. A network positioning system for the internet. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '04, pages 11–11, Berkeley, CA, USA, 2004. USENIX Association.

[77] p2psim. A simulator for peer-to-peer protocols, 2005. http://pdos.csail.mit.edu/p2psim/.

[78] Vinay Pai, Kapil Kumar, Karthik Tamilmani, Vinay Sambamurthy, and Alexander E. Mohr. Chainsaw: eliminating trees from overlay multicast. In *Proceedings of the 4th international conference on Peer-to-Peer Systems*, IPTPS'05, pages 127–140, Berlin, Heidelberg, 2005. Springer-Verlag.

[79] Roberto Perdisci, Davide Ariu, Prahlad Fogla, Giorgio Giacinto, and Wenke Lee. Mcpad: A multiple classifier system for accurate payload-based anomaly detection. *Comput. Netw.*, 53(6):864–881, April 2009.

[80] Marcelo Pias, Jon Crowcroft, Steve R. Wilbur, Timothy L. Harris, and Saleem N. Bhatti. Lighthouses for scalable distributed location. In *IPTPS*, volume 2735 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 2003.

[81] Cisco Press. Cisco visual networking index: Forecast and methodology, 2010-2015. White Paper.

[82] Pyxida. An open source network coordinate library and application. `http://pyxida.sourceforge.net`.

[83] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[84] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, pages 161–172, New York, NY, USA, 2001. ACM.

[85] Martin Rehak, Michal Pechoucek, Martin Grill, Jan Stiborek, and Karel Bartos. Game theoretical adaptation model for intrusion detection system. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '11, pages 1123–1124, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems.

[86] Sankardas Roy, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, Vivek Shandilya, and Qishi Wu. A survey of game theory as applied to network security. In *Proceedings of the 2010 43rd Hawaii International Conference on System Sciences*, HICSS '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

[87] Benjamin I.P. Rubinstein, Blaine Nelson, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J. D. Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 1–14, New York, NY, USA, 2009. ACM.

[88] Damien Saucez, Benoit Donnet, and Olivier Bonaventure. A reputation-based approach for securing vivaldi embedding system. In *Proceedings of the 13th open European summer school and IFIP TC6.6 conference on Dependable and adaptable networks and services*, EUNICE'07, pages 78–85, Berlin, Heidelberg, 2007. Springer-Verlag.

[89] Hendrik Schulze and Klaus Mochalski. Internet study 2007. `http://www.ipoque.com/en/resources/internet-studies`.

[90] Hendrik Schulze and Klaus Mochalski. Internet study 2008/2009. `http://www.ipoque.com/en/resources/internet-studies`.

[91] Jeff Seibert, Sheila Becker, Cristina Nita-Rotaru, and Radu State. Newton meets vivaldi: Using physical laws to secure virtual coordinate systems (invited abstract). In *Proceedings of the Network and Distributed System Security Symposium*, NDSS'12, pages 1–1. The Internet Society, 2012.

[92] Jeff Seibert, Sheila Becker, Cristina Nita-Rotaru, and Radu State. Securing virtual coordinates by enforcing physical laws. In *32nd International Conference on Distributed Computing Systems*, ICDCS'12, pages 1–10. IEEE, 2012.

[93] Yuval Shavitt and Tomer Tankel. On the curvature of the internet and its usage for overlay construction and distance estimation. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, page 384, March 2004.

[94] Micah Sherr, Matt Blaze, and Boon Thau Loo. Veracity: practical secure network coordinates via vote-based agreements. In *Proceedings of the 2009 conference on USENIX Annual technical conference*, USENIX'09, pages 1–14, Berkeley, CA, USA, 2009. USENIX Association.

[95] Micah Sherr, Boon Thau Loo, and Matt Blaze. Veracity: a fully decentralized service for securing network coordinate systems. In *Proceedings of the 7th international conference on Peer-to-peer systems*, IPTPS'08, pages 15–15, Berkeley, CA, USA, 2008. USENIX Association.

[96] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 305–316, Washington, DC, USA, 2010. IEEE Computer Society.

[97] Moritz Steiner and Ernst W. Biersack. Where is my peer? evaluation of the vivaldi network coordinate system in azureus. In *Proceedings of the 8th International IFIP-TC 6 Networking Conference*, NETWORKING '09, pages 145–156, Berlin, Heidelberg, 2009. Springer-Verlag.

[98] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, pages 149–160, New York, NY, USA, 2001. ACM.

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

[100] George Theodorakopoulos and John S. Baras. Game theoretic modeling of malicious users in collaborative networks. *IEEE Journal on Selected Areas in Communications*, 26(7):1317–1327, 2008.

[101] Tor-project. `https://www.torproject.org/about/overview.html.en`.

[102] V Vapnik and A Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24(6):774–780, 1963.

[103] Fernando Vega-Redondo. *Economics and the theory of games.* Cambridge University Press, 2003.

[104] John Von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[105] Gérard Wagener. *Self-Adaptive Honeypots Coercing and Assessing Attacker Behaviour.* These, Institut National Polytechnique de Lorraine - INPL, June 2011. Thèse en co-tutelle entre l'INPL et l'Université du Luxembourg sous la direction Commune de Thomas Engel et Olivier Festor avec la participation de Radu State.

[106] Gérard Wagener, Radu State, Alexandre Dulaunoy, and Thomas Engel. Self adaptive high interaction honeypots driven by game theory. In *Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, SSS '09, pages 741–755, Berlin, Heidelberg, 2009. Springer-Verlag.

[107] Cynthia Wagner, Jérôme François, Radu State, and Thomas Engel. Machine learning approach for ip-flow record anomaly detection. In *Networking (1)*, volume 6640 of *Lecture Notes in Computer Science*, pages 28–39. Springer, 2011.

[108] Cynthia Wagner, Gerard Wagener, Radu State, Thomas Engel, and Alexandre Dulaunoy. Breaking tor anonymity with game theory and data mining. In *Proceedings of the 2010 Fourth International Conference on Network and System Security*, NSS '10, pages 47–54, Washington, DC, USA, 2010. IEEE Computer Society.

[109] Guohui Wang and T.S. Eugene Ng. Distributed algorithms for stable and secure network coordinates. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, IMC '08, pages 131–144, New York, NY, USA, 2008. ACM.

[110] David Zage and Cristina Nita-Rotaru. Robust decentralized virtual coordinate systems in adversarial environments. *ACM Trans. Inf. Syst. Secur.*, 13(4):38:1–38:34, December 2010.

[111] David John Zage and Cristina Nita-Rotaru. On the accuracy of decentralized virtual coordinate systems in adversarial networks. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 214–224, New York, NY, USA, 2007. ACM.

Part IV

Appendices

Appendix A

Analyzing Attacks and Defenses on VCS

Table A.1: Abbreviations used in Appendix A.1

Abbreviation	*Term*
% Mal. Nodes	Percent of Malicious Nodes in Topology
5^{th} Prediction	5^{th} Percentile Prediction Error (Microseconds)
50^{th} Prediction	50^{th} Percentile Prediction Error (Microseconds)
95^{th} Prediction	95^{th} Percentile Prediction Error (Microseconds)
5^{th} Relative	5^{th} Percentile Relative Error
50^{th} Relative	50^{th} Percentile Relative Error
95^{th} Relative	95^{th} Percentile Relative Error
spat-temp	spatio-temporal outlier detection
spatial	spatial outlier detection
temporal	temporal outlier detection

A.1 Simulation Results

In this appendix, we present the raw simulation results for the prediction error and relative error of Vivaldi running over the King and AMP topologies under the three different attacks using different defenses techniques and thresholds. All prediction error measurement are given in microseconds. Table A.1 provides a listing of the abbreviations that are used.

Table A.2: King Data Sets Results Using a Spatial Outlier Threshold of 1.25

% Mal. Nodes	Attacker Strategy	Defender Strategy	5th prediction	50th prediction	95th prediction	5th relative	50th relative	95th relative
0	none	none	1464.8	16933.3	103225.7	1.00	1.00	1.00
10	inflation	none	3659.1	40481.8	166147.0	2.50	2.39	1.61
10	deflation	none	1637.2	18854.1	105578.8	1.12	1.11	1.02
10	oscillation	none	3445.3	39140.9	162660.8	2.35	2.31	1.58
20	inflation	none	5918.2	63977.6	217610.4	4.04	3.78	2.11
20	deflation	none	1785.5	20392.8	109044.3	1.22	1.20	1.06
20	oscillation	none	5340.7	65438.2	220359.7	3.65	3.86	2.13
30	inflation	none	7321.1	84461.4	260568.9	5.00	4.99	2.52
30	deflation	none	2110.9	24051.6	122280.1	1.44	1.42	1.18
30	oscillation	none	6203.1	82698.8	271028.3	4.23	4.88	2.63
10	deflation	spat-temp	1370.9	15953.9	98979.3	0.94	0.94	0.96
20	deflation	spat-temp	1419.8	16481.8	99997.0	0.97	0.97	0.97
30	deflation	spat-temp	1623.0	18791.0	108166.9	1.11	1.11	1.05
10	deflation	spatial	1340.9	15715.7	104368.5	0.92	0.93	1.01
20	deflation	spatial	1410.0	16346.7	97992.8	0.96	0.97	0.95
30	deflation	spatial	1592.7	18448.2	108813.4	1.09	1.09	1.05
10	deflation	temporal	1573.4	18123.1	103729.8	1.07	1.07	1.00
20	deflation	temporal	1801.4	20683.0	111640.1	1.23	1.22	1.08
30	deflation	temporal	2047.5	23481.1	124543.8	1.40	1.39	1.21
10	oscillation	spat-temp	1463.2	17103.4	108593.3	1.00	1.01	1.05
20	oscillation	spat-temp	1458.6	17264.5	111034.1	1.00	1.02	1.08
30	oscillation	spat-temp	2329.3	28450.4	161415.0	1.59	1.68	1.56
10	oscillation	spatial	1462.8	17093.5	111463.2	1.00	1.01	1.08
20	oscillation	spatial	1432.8	17057.7	109230.9	0.98	1.01	1.06
30	oscillation	spatial	2394.2	29371.6	163011.2	1.63	1.73	1.58
10	oscillation	temporal	2974.9	34233.2	153880.4	2.03	2.02	1.49
20	oscillation	temporal	4556.6	54988.6	198632.5	3.11	3.25	1.92
30	oscillation	temporal	5801.0	75013.4	253542.8	3.96	4.43	2.46
10	inflation	spat-temp	1500.9	17698.7	115869.1	1.02	1.05	1.12
20	inflation	spat-temp	1945.9	23773.4	143452.8	1.33	1.40	1.39
30	inflation	spat-temp	2849.9	36814.8	203761.1	1.95	2.17	1.97
10	inflation	spatial	1484.8	17523.1	117843.0	1.01	1.03	1.14
20	inflation	spatial	1958.0	24094.4	152662.2	1.34	1.42	1.48
30	inflation	spatial	2921.1	37996.7	202977.7	1.99	2.24	1.97
10	inflation	temporal	3463.6	39203.1	165743.9	2.36	2.32	1.61
20	inflation	temporal	5333.6	59503.5	204578.2	3.64	3.51	1.98
30	inflation	temporal	7273.5	84751.9	258242.9	4.97	5.01	2.50

Table A.3: King Data Sets Results Using a Spatial Outlier Threshold of 1.50

% Mal. Nodes	Attacker Strategy	Defender Strategy	5th prediction	50th prediction	95th prediction	5th relative	50th relative	95th relative
0	none	none	1464.8	16933.3	103225.7	1.00	1.00	1.00
10	inflation	none	3659.1	40481.8	166147.0	2.50	2.39	1.61
10	deflation	none	1637.2	18854.1	105578.8	1.12	1.11	1.02
10	oscillation	none	3445.3	39140.9	162660.8	2.35	2.31	1.58
20	inflation	none	5918.2	63977.6	217610.4	4.04	3.78	2.11
20	deflation	none	1785.5	20392.8	109044.3	1.22	1.20	1.06
20	oscillation	none	5340.7	65438.2	220359.7	3.65	3.86	2.13
30	inflation	none	7321.1	84461.4	260568.9	5.00	4.99	2.52
30	deflation	none	2110.9	24051.6	122280.1	1.44	1.42	1.18
30	oscillation	none	6203.1	82698.8	271028.3	4.23	4.88	2.63
10	oscillation	spat-temp	1415.2	16483.0	102983.0	0.97	0.97	1.00
20	oscillation	spat-temp	1865.2	22647.5	136199.0	1.27	1.34	1.32
30	oscillation	spat-temp	2940.6	35349.6	172645.7	2.01	2.09	1.67
10	oscillation	spatial	1425.6	16660.2	103692.1	0.97	0.98	1.00
20	oscillation	spatial	1924.8	23462.4	134401.5	1.31	1.39	1.30
30	oscillation	spatial	3020.3	36376.6	178286.5	2.06	2.15	1.73
10	oscillation	temporal	2778.4	32305.8	150138.3	1.90	1.91	1.45
20	oscillation	temporal	4733.0	56431.4	203950.7	3.23	3.33	1.98
30	oscillation	temporal	5768.1	73822.3	249418.3	3.94	4.36	2.42
10	deflation	spat-temp	1383.5	16051.9	98864.3	0.94	0.95	0.96
20	deflation	spat-temp	1490.7	17298.6	102227.4	1.02	1.02	0.99
30	deflation	spat-temp	1809.6	21048.8	120365.0	1.24	1.24	1.17
10	deflation	spatial	1354.1	15712.4	96567.1	0.92	0.93	0.94
20	deflation	spatial	1494.8	17246.3	102603.2	1.02	1.02	0.99
30	deflation	spatial	1754.1	20215.6	113067.5	1.20	1.19	1.10
10	deflation	temporal	1570.1	18038.9	104270.1	1.07	1.07	1.01
20	deflation	temporal	1892.3	21702.0	114209.6	1.29	1.28	1.11
30	deflation	temporal	2055.0	23581.2	122008.4	1.40	1.39	1.18
10	inflation	spat-temp	1705.8	20294.1	124540.2	1.16	1.20	1.21
20	inflation	spat-temp	2352.7	29360.2	168829.9	1.61	1.73	1.64
30	inflation	spat-temp	4268.7	53626.2	225033.4	2.91	3.17	2.18
10	inflation	spatial	1667.3	19750.4	127066.2	1.14	1.17	1.23
20	inflation	spatial	2396.6	29820.9	165766.3	1.64	1.76	1.61
30	inflation	spatial	4112.2	51677.4	222180.9	2.81	3.05	2.15
10	inflation	temporal	3417.8	38647.1	160660.4	2.33	2.28	1.56
20	inflation	temporal	5826.0	64156.2	215632.6	3.98	3.79	2.09
30	inflation	temporal	7295.4	84753.8	259427.8	4.98	5.01	2.51

Table A.4: King Data Sets Results Using a Spatial Outlier Threshold of 1.75

% Mal. Nodes	Attacker Strategy	Defender Strategy	5th prediction	50th prediction	95th prediction	5th relative	50th relative	95th relative
0	none	none	1464.8	16933.3	103225.7	1.00	1.00	1.00
10	inflation	none	3659.1	40481.8	166147.0	2.50	2.39	1.61
10	deflation	none	1637.2	18854.1	105578.8	1.12	1.11	1.02
10	oscillation	none	3445.3	39140.9	162660.8	2.35	2.31	1.58
20	inflation	none	5918.2	63977.6	217610.4	4.04	3.78	2.11
20	deflation	none	1785.5	20392.8	109044.3	1.22	1.20	1.06
20	oscillation	none	5340.7	65438.2	220359.7	3.65	3.86	2.13
30	inflation	none	7321.1	84461.4	260568.9	5.00	4.99	2.52
30	deflation	none	2110.9	24051.6	122280.1	1.44	1.42	1.18
30	oscillation	none	6203.1	82698.8	271028.3	4.23	4.88	2.63
10	deflation	spat-temp	1435.7	16746.5	102636.2	0.98	0.99	0.99
20	deflation	spat-temp	1652.1	19188.0	110804.1	1.13	1.13	1.07
30	deflation	spat-temp	1893.9	21855.0	118357.6	1.29	1.29	1.15
10	deflation	spatial	1413.1	16363.3	98436.6	0.96	0.97	0.95
20	deflation	spatial	1633.3	19030.1	110567.0	1.11	1.12	1.07
30	deflation	spatial	1853.6	21308.5	114327.0	1.27	1.26	1.11
10	deflation	temporal	1575.9	18181.6	105624.5	1.08	1.07	1.02
20	deflation	temporal	1811.3	20858.8	115762.2	1.24	1.23	1.12
30	deflation	temporal	2019.7	23145.0	121965.9	1.38	1.37	1.18
10	oscillation	spat-temp	1389.3	16297.2	105217.6	0.95	0.96	1.02
20	oscillation	spat-temp	2268.2	27149.6	139436.6	1.55	1.60	1.35
30	oscillation	spat-temp	3837.3	46193.0	188223.1	2.62	2.73	1.82
10	oscillation	spatial	1407.1	16553.3	103825.4	0.96	0.98	1.01
20	oscillation	spatial	2227.3	26858.6	145171.2	1.52	1.59	1.41
30	oscillation	spatial	3785.5	46098.0	190703.2	2.58	2.72	1.85
10	oscillation	temporal	2754.9	31836.6	143407.5	1.88	1.88	1.39
20	oscillation	temporal	4330.9	51601.4	190480.0	2.96	3.05	1.85
30	oscillation	temporal	5359.5	68300.9	232029.6	3.66	4.03	2.25
10	inflation	spat-temp	1785.7	21433.6	125352.3	1.22	1.27	1.21
20	inflation	spat-temp	2893.7	35984.5	172251.7	1.98	2.13	1.67
30	inflation	spat-temp	5555.8	66008.9	226387.8	3.79	3.90	2.19
10	inflation	spatial	1731.2	20857.7	123971.6	1.18	1.23	1.20
20	inflation	spatial	2727.4	34103.9	168721.0	1.86	2.01	1.63
30	inflation	spatial	5859.0	69282.7	224944.3	4.00	4.09	2.18
10	inflation	temporal	3369.7	38104.0	155036.6	2.30	2.25	1.50
20	inflation	temporal	5457.8	61362.0	208658.5	3.73	3.62	2.02
30	inflation	temporal	6889.1	79410.5	241237.6	4.70	4.69	2.34

Table A.5: AMP Data Sets Results Using a Spatial Outlier Threshold of 1.25

% Mal. Nodes	Attacker Strategy	Defender Strategy	5th prediction	50th prediction	95th prediction	5th relative	50th relative	95th relative
0	none	none	1410.2	16754.3	125433.2	1.00	1.00	1.00
10	inflation	none	2689.6	31586.7	160820.3	1.91	1.89	1.28
10	deflation	none	1586.1	18572.8	123041.0	1.12	1.11	0.98
10	oscillation	none	2752.2	30505.7	149437.5	1.95	1.82	1.19
20	inflation	none	3850.7	41202.6	173186.5	2.73	2.46	1.38
20	deflation	none	1830.2	20436.6	123122.9	1.30	1.22	0.98
20	oscillation	none	3996.4	45351.5	179879.8	2.83	2.71	1.43
30	inflation	none	4761.6	49206.7	188199.9	3.38	2.94	1.50
30	deflation	none	2096.1	23484.8	126563.0	1.49	1.40	1.01
30	oscillation	none	4833.9	51360.0	207902.3	3.43	3.07	1.66
10	deflation	spat-temp	1404.8	17058.2	130853.2	1.00	1.02	1.04
20	deflation	spat-temp	1393.1	17357.0	132062.9	0.99	1.04	1.05
30	deflation	spat-temp	1503.3	17667.9	131765.6	1.07	1.05	1.05
10	deflation	spatial	1306.0	16140.2	130211.3	0.93	0.96	1.04
20	deflation	spatial	1362.9	16473.7	126121.1	0.97	0.98	1.01
30	deflation	spatial	1485.8	17820.6	130912.7	1.05	1.06	1.04
10	deflation	temporal	1611.9	18762.2	122760.3	1.14	1.12	0.98
20	deflation	temporal	1781.4	20632.7	135162.2	1.26	1.23	1.08
30	deflation	temporal	1954.5	23069.6	128202.8	1.39	1.38	1.02
10	oscillation	spat-temp	1355.3	16348.9	129880.2	0.96	0.98	1.04
20	oscillation	spat-temp	1493.7	18080.8	128506.8	1.06	1.08	1.02
30	oscillation	spat-temp	1949.7	23989.9	147526.3	1.38	1.43	1.18
10	oscillation	spatial	1485.7	17385.6	139967.4	1.05	1.04	1.12
20	oscillation	spatial	1496.9	17582.0	134691.7	1.06	1.05	1.07
30	oscillation	spatial	2316.7	30238.9	164896.8	1.64	1.80	1.31
10	oscillation	temporal	2227.8	24861.2	139700.2	1.58	1.48	1.11
20	oscillation	temporal	3412.8	37701.9	165242.1	2.42	2.25	1.32
30	oscillation	temporal	4358.2	48582.6	191376.7	3.09	2.90	1.53
10	inflate	spat-temp	1214.0	15583.1	130982.6	0.86	0.93	1.04
20	inflate	spat-temp	1791.7	24350.4	153215.5	1.27	1.45	1.22
30	inflate	spat-temp	3626.2	43913.9	192297.6	2.57	2.62	1.53
10	inflate	spatial	1323.0	16321.0	135088.6	0.94	0.97	1.08
20	inflate	spatial	1558.6	21180.6	151849.3	1.11	1.26	1.21
30	inflate	spatial	2418.1	31925.8	175905.5	1.71	1.91	1.40
10	inflate	temporal	1777.6	20551.1	137741.7	1.26	1.23	1.10
20	inflate	temporal	3392.2	38282.3	174256.7	2.41	2.28	1.39
30	inflate	temporal	4722.6	50453.4	194122.4	3.35	3.01	1.55

Table A.6: AMP Data Sets Results Using a Spatial Outlier Threshold of 1.50

% Mal. Nodes	Attacker Strategy	Defender Strategy	5^{th} prediction	50^{th} prediction	95^{th} prediction	5^{th} relative	50^{th} relative	95^{th} relative
0	none	none	1410.2	16754.3	125433.2	1.00	1.00	1.00
10	inflation	none	2689.6	31586.7	160820.3	1.91	1.89	1.28
10	deflation	none	1586.1	18572.8	123041.0	1.12	1.11	0.98
10	oscillation	none	2752.2	30505.7	149437.5	1.95	1.82	1.19
20	inflation	none	3850.7	41202.6	173186.5	2.73	2.46	1.38
20	deflation	none	1830.2	20436.6	123122.9	1.30	1.22	0.98
20	oscillation	none	3996.4	45351.5	179879.8	2.83	2.71	1.43
30	inflation	none	4761.6	49206.7	188199.9	3.38	2.94	1.50
30	deflation	none	2096.1	23484.8	126563.0	1.49	1.40	1.01
30	oscillation	none	4833.9	51360.0	207902.3	3.43	3.07	1.66
10	deflation	spat-temp	1404.8	17058.2	130853.2	1.00	1.02	1.04
20	deflation	spat-temp	1393.1	17357.0	132062.9	0.99	1.04	1.05
30	deflation	spat-temp	1503.3	17667.9	131765.6	1.07	1.05	1.05
10	deflation	spatial	1306.0	16140.2	130211.3	0.93	0.96	1.04
20	deflation	spatial	1362.9	16473.7	126121.1	0.97	0.98	1.01
30	deflation	spatial	1485.8	17820.6	130912.7	1.05	1.06	1.04
10	deflation	temporal	1611.9	18762.2	122760.3	1.14	1.12	0.98
20	deflation	temporal	1781.4	20632.7	135162.2	1.26	1.23	1.08
30	deflation	temporal	1954.5	23069.6	128202.8	1.39	1.38	1.02
10	oscillation	spat-temp	1355.3	16348.9	129880.2	0.96	0.98	1.04
20	oscillation	spat-temp	1493.7	18080.8	128506.8	1.06	1.08	1.02
30	oscillation	spat-temp	1949.7	23989.9	147526.3	1.38	1.43	1.18
10	oscillation	spatial	1485.7	17385.6	139967.4	1.05	1.04	1.12
20	oscillation	spatial	1496.9	17582.0	134691.7	1.06	1.05	1.07
30	oscillation	spatial	2316.7	30238.9	164896.8	1.64	1.80	1.31
10	oscillation	temporal	2227.8	24861.2	139700.2	1.58	1.48	1.11
20	oscillation	temporal	3412.8	37701.9	165242.1	2.42	2.25	1.32
30	oscillation	temporal	4358.2	48582.6	191376.7	3.09	2.90	1.53
10	inflate	spat-temp	1214.0	15583.1	130982.6	0.86	0.93	1.04
20	inflate	spat-temp	1791.7	24350.4	153215.5	1.27	1.45	1.22
30	inflate	spat-temp	3626.2	43913.9	192297.6	2.57	2.62	1.53
10	inflate	spatial	1323.0	16321.0	135088.6	0.94	0.97	1.08
20	inflate	spatial	1558.6	21180.6	151849.3	1.11	1.26	1.21
30	inflate	spatial	2418.1	31925.8	175905.5	1.71	1.91	1.40
10	inflate	temporal	1777.6	20551.1	137741.7	1.26	1.23	1.10
20	inflate	temporal	3392.2	38282.3	174256.7	2.41	2.28	1.39
30	inflate	temporal	4722.6	50453.4	194122.4	3.35	3.01	1.55

Table A.7: AMP Data Sets Results Using a Spatial Outlier Threshold of 1.75

% Mal. Nodes	Attacker Strategy	Defender Strategy	5th prediction	50th prediction	95th prediction	5th relative	50th relative	95th relative
0	none	none	1410.2	16754.3	125433.2	1.00	1.00	1.00
10	inflation	none	2689.6	31586.7	160820.3	1.91	1.89	1.28
10	deflation	none	1586.1	18572.8	123041.0	1.12	1.11	0.98
10	oscillation	none	2752.2	30505.7	149437.5	1.95	1.82	1.19
20	inflation	none	3850.7	41202.6	173186.5	2.73	2.46	1.38
20	deflation	none	1830.2	20436.6	123122.9	1.30	1.22	0.98
20	oscillation	none	3996.4	45351.5	179879.8	2.83	2.71	1.43
30	inflation	none	4761.6	49206.7	188199.9	3.38	2.94	1.50
30	deflation	none	2096.1	23484.8	126563.0	1.49	1.40	1.01
30	oscillation	none	4833.9	51360.0	207902.3	3.43	3.07	1.66
10	deflation	spat-temp	1430.7	17202.8	129255.3	1.01	1.03	1.03
20	deflation	spat-temp	1609.1	19146.1	129063.5	1.14	1.14	1.03
30	deflation	spat-temp	1759.2	20681.3	132526.2	1.25	1.23	1.06
10	deflation	spatial	1343.7	16738.8	127774.9	0.95	1.00	1.02
20	deflation	spatial	1416.7	17853.3	128162.9	1.00	1.07	1.02
30	deflation	spatial	1842.1	20699.6	128417.8	1.31	1.24	1.02
10	deflation	temporal	1565.2	18856.6	127559.9	1.11	1.13	1.02
20	deflation	temporal	1636.8	19989.0	127893.4	1.16	1.19	1.02
30	deflation	temporal	1827.8	21078.2	127307.2	1.30	1.26	1.01
10	oscillation	spat-temp	1422.6	17165.2	130704.0	1.01	1.02	1.04
20	oscillation	spat-temp	2062.6	24405.2	144747.6	1.46	1.46	1.15
30	oscillation	spat-temp	3494.2	38316.6	165713.3	2.48	2.29	1.32
10	oscillation	spatial	1355.9	16846.3	127641.4	0.96	1.01	1.02
20	oscillation	spatial	2005.0	24366.4	139735.4	1.42	1.45	1.11
30	oscillation	spatial	2809.9	31204.1	156410.3	1.99	1.86	1.25
10	oscillation	temporal	2233.8	25638.3	143041.7	1.58	1.53	1.14
20	oscillation	temporal	3319.3	36426.9	167520.5	2.35	2.17	1.34
30	oscillation	temporal	4163.7	45697.6	186174.1	2.95	2.73	1.48
10	inflate	spat-temp	1395.9	18046.9	138020.7	0.99	1.08	1.10
20	inflate	spat-temp	2860.4	32648.6	156973.7	2.03	1.95	1.25
30	inflate	spat-temp	3581.2	39709.9	179662.9	2.54	2.37	1.43
10	inflate	spatial	1688.8	21131.1	145696.2	1.20	1.26	1.16
20	inflate	spatial	2181.4	26812.6	156035.5	1.55	1.60	1.24
30	inflate	spatial	4284.4	47074.1	177853.2	3.04	2.81	1.42
10	inflate	temporal	1870.6	22213.2	139487.2	1.33	1.33	1.11
20	inflate	temporal	3844.2	42157.5	177436.3	2.73	2.52	1.41
30	inflate	temporal	5084.5	53068.7	197140.3	3.61	3.17	1.57

Table A.8: AMP Data Sets Results Using a Spatial Outlier Threshold of 2.00

% Mal. Nodes	Attacker Strategy	Defender Strategy	5th prediction	50th prediction	95th prediction	5th relative	50th relative	95th relative
0	none	none	1410.2	16754.3	125433.2	1.00	1.00	1.00
10	inflation	none	2689.6	31586.7	160820.3	1.91	1.89	1.28
10	deflation	none	1586.1	18572.8	123041.0	1.12	1.11	0.98
10	oscillation	none	2752.2	30505.7	149437.5	1.95	1.82	1.19
20	inflation	none	3850.7	41202.6	173186.5	2.73	2.46	1.38
20	deflation	none	1830.2	20436.6	123122.9	1.30	1.22	0.98
20	oscillation	none	3996.4	45351.5	179879.8	2.83	2.71	1.43
30	inflation	none	4761.6	49206.7	188199.9	3.38	2.94	1.50
30	deflation	none	2096.1	23484.8	126563.0	1.49	1.40	1.01
30	oscillation	none	4833.9	51360.0	207902.3	3.43	3.07	1.66
10	deflation	spat-temp	1464.4	18419.5	130288.7	1.04	1.10	1.04
20	deflation	spat-temp	1662.7	19132.8	129887.5	1.18	1.14	1.04
30	deflation	spat-temp	1607.5	19504.8	127343.1	1.14	1.16	1.02
10	deflation	spatial	1554.0	18654.4	137540.5	1.10	1.11	1.10
20	deflation	spatial	1615.4	19105.1	132091.8	1.15	1.14	1.05
30	deflation	spatial	1820.8	20546.6	131763.7	1.29	1.23	1.05
10	deflation	temporal	1622.1	18825.3	122979.3	1.15	1.12	0.98
20	deflation	temporal	1665.2	19946.4	129018.8	1.18	1.19	1.03
30	deflation	temporal	1920.6	22624.1	127540.0	1.36	1.35	1.02
10	oscillation	spat-temp	1235.0	15871.9	128051.1	0.88	0.95	1.02
20	oscillation	spat-temp	2195.8	26098.6	142114.6	1.56	1.56	1.13
30	oscillation	spat-temp	4101.5	44928.3	189272.2	2.91	2.68	1.51
10	oscillation	spatial	1391.1	16449.7	129586.5	0.99	0.98	1.03
20	oscillation	spatial	1733.7	20956.1	137333.2	1.23	1.25	1.09
30	oscillation	spatial	3769.9	41838.8	173715.9	2.67	2.50	1.38
10	oscillation	temporal	2071.0	24373.8	142872.5	1.47	1.45	1.14
20	oscillation	temporal	3460.5	37500.8	168495.2	2.45	2.24	1.34
30	oscillation	temporal	4515.3	49473.3	207819.2	3.20	2.95	1.66
10	inflation	spat-temp	1762.9	21726.8	144118.9	1.25	1.30	1.15
20	inflation	spat-temp	3194.9	38093.6	159115.8	2.27	2.27	1.27
30	inflation	spat-temp	4685.9	49816.9	186829.6	3.32	2.97	1.49
10	inflation	spatial	1550.7	19507.3	142458.4	1.10	1.16	1.14
20	inflation	spatial	2859.8	35902.2	159813.0	2.03	2.14	1.27
30	inflation	spatial	4847.2	50995.2	201821.5	3.44	3.04	1.61
10	inflation	temporal	1699.1	21003.3	137749.2	1.20	1.25	1.10
20	inflation	temporal	4169.7	42674.4	169818.6	2.96	2.55	1.35
30	inflation	temporal	5512.9	59406.6	214344.8	3.91	3.55	1.71