EVOLUTIONARY SOLUTIONS AND INTERNET APPLICATIONS FOR ALGORITHMIC GAME THEORY

by

Christine Chung

B.A. in Computer Science, Cornell University, 1999

M.Eng. in Computer Science, Cornell University, 2000

Submitted to the Graduate Faculty of

Arts and Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2009

UNIVERSITY OF PITTSBURGH

ARTS AND SCIENCES

This dissertation was presented

by

Christine Chung

It was defended on

June 10, 2009

and approved by

Kirk Pruhs, Professor, Department of Computer Science

Panos Chrysanthis, Professor, Department of Computer Science

Alexandros Labrinidis, Associate Professor, Department of Computer Science

Avrim Blum, Professor, Department of Computer Science, Carnegie Mellon University

Dissertation Director: Kirk Pruhs, Professor, Department of Computer Science

EVOLUTIONARY SOLUTIONS AND INTERNET APPLICATIONS FOR ALGORITHMIC GAME THEORY

Christine Chung, PhD

University of Pittsburgh, 2009

The growing pervasiveness of the internet has created a new class of algorithmic problems: those in which the strategic interaction of autonomous, self-interested entities must be accounted for. So motivated, we seek to (1) use game theoretic models and techniques to study practical problems in load balancing, data streams and internet traffic congestion, and (2) demonstrate the usefulness of evolutionary game theory's adaptive learning model as an analytical and evaluative tool.

First we consider the evolutionary game theory concept of stochastic stability, and propose the *price of stochastic anarchy* as an alternative to the *price of anarchy* for quantifying the cost of having no central authority. Unlike Nash equilibria, stochastically stable states are the result of natural dynamics of large populations of computationally bounded agents, and are resilient to small perturbations from ideal play. To illustrate the utility of stochastic stability, we study the load balancing game on related machines, which has an unbounded price of anarchy, even in the case of two jobs and two machines. We show that in contrast, even in the general case, the price of stochastic anarchy is bounded.

Next, we propose auction-based mechanisms for admission control of continuous queries to a Data Stream Management System. When submitting a query, each user also submits a bid: how much she is willing to pay for her query to run. Our mechanisms must admit queries and set payments in a way that maximizes system revenue while incentivizing customers to use the system honestly. We propose several manipulation-resistant payment mechanisms and prove that one guarantees a profit close to a standard profit benchmark, and the others

perform well experimentally.

Finally, we study the long standing problem of congestion control at bottleneck routers on the internet. We examine the effectiveness of commonly-used queuing policies when each network endpoint is self-interested and has no information about the other endpoints' actions or preferences. By employing evolutionary game theory, we find that while bottleneck routers face heavy congestion at stochastically stable states under policies being currently deployed, a practical policy that was recently proposed yields fair and efficient conditions with no congestion.

TABLE OF CONTENTS

PREFACE x

1.0 INTRODUCTION 1

1.1 Quantifying Inefficiency of Equilibria 5

1.1.1 Our Contribution: the Price of Stochastic Anarchy 7

1.2 Algorithmic Mechanism Design 9

1.2.1 Our Contribution: Data Streams Query Admission 10

1.3 Congestion at Internet Bottleneck Routers 12

2.0 THE PRICE OF STOCHASTIC ANARCHY 14

2.1 Introduction 14

2.1.1 Our Results 15

2.1.2 Related Work 16

2.2 Model and Background 18

2.2.1 Adaptive Play and Stochastic Stability 18

2.2.2 Imitation Dynamics 21

2.3 Load Balancing: Game Definition and Price of Nash Anarchy 22

2.4 Upper Bound on Price of Stochastic Anarchy 24

2.4.1 Two Players, Two Machines 24

2.4.2 General Case: n Players, m Machines 27

2.5 Lower Bound on Price of Stochastic Anarchy 30

2.6 Summary 33

3.0 CONTINUOUS QUERY ADMISSION CONTROL 35

3.1 Introduction 35

3.1.1 System Model and Problem Statement . . . 37
3.1.2 Relevant Background on Auctions . . . 40
3.1.3 This Work . . . 42
3.2 Greedy Strategyproof Mechanisms . . . 45
3.2.1 Agents Chosen by Remaining Load . . . 45
3.2.2 Agents Chosen by Static Fair Share Load . . . 47
3.2.2.1 CAF (CQ Admission based on Fair Share) . . . 47
3.2.2.2 CAF+: An Extension to CAF . . . 48
3.2.3 Agents Chosen by Total Load . . . 50
3.3 A Profit Guarantee . . . 51
3.3.1 A Randomized Mechanism . . . 51
3.3.1.1 Strategyproofness . . . 52
3.3.1.2 Competitiveness . . . 54
3.4 Sybil Attack . . . 56
3.4.1 Attacks Against the Fair Share Mechanisms . . . 57
3.4.2 Attacks Against the Total Load Mechanisms . . . 58
3.4.3 Attacks Against the Randomized Mechanism . . . 61
3.5 Experimental Evaluation . . . 63
3.5.1 Experimental Platform . . . 63
3.5.2 Experimental Results. . . . 65
3.6 Summary . . . 69
4.0 INTERNET BOTTLENECK ROUTER CONGESTION . . . 73
4.1 Introduction . . . 73
4.1.1 Our Results . . . 75
4.1.2 Related Work . . . 77
4.2 Model, Notation, and Background . . . 79
4.2.1 Adaptive Learning and Imitation Dynamics . . . 79
4.3 Droptail . . . 81
4.3.1 Droptail's Best Response . . . 82
4.3.2 Droptail's NE . . . 83

4.3.3 Droptail's Stochastically Stable States 85
4.4 RED (Random Early Detection) 88
4.4.1 RED's Best Response 90
4.4.2 RED's Nash Equilibria 93
4.4.3 RED's Stochastically Stable States 97
4.5 "Fair" Queue Policy 100
4.6 Summary 103
5.0 CONCLUSIONS 106
BIBLIOGRAPHY 109

LIST OF TABLES

1 Summary of the greedy algorithms 43
2 Properties of our proposed auction mechanisms 45
3 A sybil attack against CAT+. 59
4 Workload Characteristics 64
5 Runtime Performance. 68
6 Properties of our proposed auction mechanisms. 70

LIST OF FIGURES

1 Research goals. . . . 4
2 Unbounded price of anarchy in a load balancing game. . . . 8
3 A load-balancing game with bad price of stochastic anarchy. . . . 30
4 PSA can be worse than m. . . . 33
5 Two views of a sample input instance. . . . 38
6 A Sybil Attack. . . . 56
7 Experimental results: admission rate, profit and payoff. . . . 66
8 Experimental results for profit as capacity varies. . . . 68
9 Profit comparison. . . . 70
10 Ranges of g values for three different RED best response functions. . . . 91

PREFACE

First, I would like to thank my adviser, Kirk Pruhs, for his invaluable guidance, for keeping me on track and pushing me in my research, for all his practical career advice, for generously supporting me and providing me with many research opportunities, and for always having my best interests in mind. I am deeply indebted to him, and I was fortunate to have found such a wonderful adviser.

I would also like to thank my committee members Panos Chrysanthis, Alex Labrinidis, and Avrim Blum. Panos and Alex have made me feel like an honorary member of their flourishing data management group, welcoming me into their ADMT family and providing me much help and guidance over the years. They have been like co-advisers to me. And Avrim, though very busy at CMU with his endless line of successful students, has generously made time in his schedule to give me advice and feedback on my research and writing.

I also feel deep gratitude for my collaborators and co-authors, from whom I have learned so much and with whom I've thoroughly enjoyed working: Katrina Ligett, Aaron Roth, Evangelia Pyrga, Lory Al Moakar, Shenoda Guirguis, Panickos Neophytou, Rob Van Stee, Giorgos Christodoulou, Alexander Souza, Tim Nonner, and Patchrawat Uthaisombut (who taught me a great deal and had endless patience when working with me on my first publication as a graduate student). You have all helped me to personally affirm that there's little in life that can be as fun and rewarding as research. Others I'd like to thank for playing a big role in my grad school experience include Michel Hanna, Peter Djalaliev, Weijia Li, Subrata Acharya, Jeff Nelson and Ricky Sanchez.

Before graduate school, I received valuable support and encouragement from my internship adviser at Bell Labs, Bob Kurshan, and my superviser at Sportvision, Rick Cavallaro.

And in college, my excitement for computer science (and related intellectual pursuits)

was inspired and nurtured by many professors (Jon Kleinberg, Eva Tardos, Bart Selman, Joe Halpern, Lillian Lee), as well as other students and friends (Rami Baalbaki, Aaron Lowenkron, Tomasz Piech, Thomas Hwang, Bob Rossi, Felix Rodriguez, Garrett Lang, Scott Aaronson) who have also influenced me in ways beyond the intellectual. Thanks also to my "family" at the Mandarin House for their love and support, and to the Cornell Women's Ultimate team (especially our coach, Mandy Carreiro), from whom I first learned true commitment, dedication, and passion.

If my interest in computer science blossomed in college, it was seeded in high school by my computer science and physics teacher, Kenneth Appel. Without him, I would never have become a computer scientist. I also need to thank my lifelong friends Felicia Yue and Abby Weintraub, who have stuck with me as our lives and careers have evolved through the years.

Finally, thank you to my family, Judy, Fung-Lung, Christopher, and Clifford, for always loving me, supporting me, humoring me, and making me laugh. You are the greatest and most special people I know! That is, besides Brian, to whom I dedicate this thesis. I don't have the words to express my abiding love and gratitude for you.

1.0 INTRODUCTION

The growing pervasiveness of the internet and its new technologies has created a new class of algorithmic problems: those in which the strategic interaction of independent, self-interested entities must be accounted for. New internet-based settings have given rise to problems where solutions are not always dictated by a central authority, but often formed by the aggregation of the actions of separate, autonomous agents.

Consider the formation of the internet itself: internet service providers (ISPs) sprang up around the world, each ISP with a set of endpoints it sought to connect in a way that was most self-serving and profitable. Thus, the topology of the internet was not designed by some central authority for the overall social good, but resulted from the aggregate decisions of independent entities whose self-interests were often in conflict with one another.

As another example, consider what happens as traffic is routed over a network. The network might be a network of computers and the traffic might be data traffic, but one might alternatively simply consider a network of roads and automobile traffic. Each car on the road wishes to travel from its starting point to its destination as quickly as possible, and routes itself accordingly. However, such separate, individual routing decisions might cause certain routes to become congested, decreasing overall travel efficiency, making the average travel time higher than if a central traffic controller dictated the routes.

We could also consider the dominant source of Google's revenue: online ad sales. Google makes money by selling ad slots for each search string entered by users of Google's free and ubiquitous search engine. Of course, Google seeks to design an auction system that incentivizes advertisers to bid their true valuation for each search string, to determine the actual market value of each ad slot. On the other hand, advertisers (companies and individuals) bidding on the search strings inevitably try to find ways to strategize and manipulate Google's

pricing system in their favor. Indeed, there is an entire industry that sells consulting services to help advertisers optimize their bidding strategies against Google's ad auction system. The prices Google charges for ad slots are a function of the bids of the many potential advertisers for each search string.

And finally, consider network endpoints behind a bottleneck router on the internet. Each endpoint makes decisions about how much traffic to send through the router, while the router, being of limited capacity, runs a protocol deciding which packets to send through and which packets to drop. Each of the endpoints is an independent agent making decisions and choosing a strategy for transmitting its packets based on its own interests, without regard to the others. The aggregate decisions of the network endpoints determine how many packets the router drops from each traffic flow. Given the fact that the bottleneck router has limited bandwidth/capacity, will greedier packet flows inevitably quash the traffic flow of the endpoints that try to send less traffic? Is there anything the designer of the router's protocol can do to prevent this?

There are innumerable computing and internet-based settings that echo the same story: self-interested, autonomous entities interact and the combination of their individual decisions together determine the final system state or outcome. These settings in fact fit the definition of a *game*: a setting where a number of decision-making agents interact, each agent (or player) with a set of possible strategies or actions to choose from, and each with a personal preference over the possible outcomes of their interaction (expressed as a *utility* or *payoff* function they wish to maximize). In the example of endpoints behind a bottleneck router, each player perhaps wishes transmit its data so as to maximize the number of packets it successfully sends minus the number of its packets that get dropped. In the case of advertiser's buying Google's ad words, each player's goal would be to place the bid that maximizes the difference between his valuation of the ad spot and the amount he must pay. In the case of traffic routing, each player chooses her own route perhaps for the goal of minimizing her travel time from starting point to destination (i.e., maximizing the negation). And in the internet formation example, each ISP's goal might be to connect its clients in a way that minimizes its own cost.

One natural question is, how bad do things actually get when strategic players trying to

maximize their own payoffs behave selfishly? For example, in the traffic routing problem, how many times worse is the average travel time when the players make their own routing choices than the minimum-possible average travel time? With internet network formation, how many times costlier is the total construction of the final network formed by the many separate ISPs than a minimum-possible-cost network?

Another natural question is how we might design underlying protocols (aka *mechanisms*) in such a way that agents' self-interests become aligned with the designer's central objectives. Google, for example, wishes to design an auction mechanism that guarantees any payoff maximizing bidder will *always* prefer to reveal his true valuation, rather than bid falsely. And protocols for bottleneck routers on the internet should be designed to incentivize network endpoints to share router capacity fairly.

These new and vital applications have fueled the growth of the field of algorithmic game theory. Algorithmic game theory is simply game theory from an algorithmic perspective: it includes algorithm design when strategic agents are part of the input with a concern for computational complexity, and proving worst-case guarantees on the performance of these algorithms. It includes evaluation of solution concepts with an emphasis on whether finding such solutions is computationally tractable. And it includes the general game theoretic study of any real-world problems that now arise in computer-based settings.

The goal of our research is to contribute to shaping and enriching the study of algorithmic game theory. (See Figure 1.) We wish to widen the breadth of understanding of game theory by introducing the application of lesser-known areas of game theory, and we also seek to supply game theoretic frameworks and analyses for a wider range of applied areas of computer science. Specifically, our goals are

1. to use game theoretic models and techniques to study practical problems in load balancing, data streams and internet traffic congestion, and
2. to demonstrate the usefulness of evolutionary game theory's adaptive learning model as an analytical and evaluative tool.

Toward these goals, we have accomplished the following, which are described in further detail later in this Chapter.

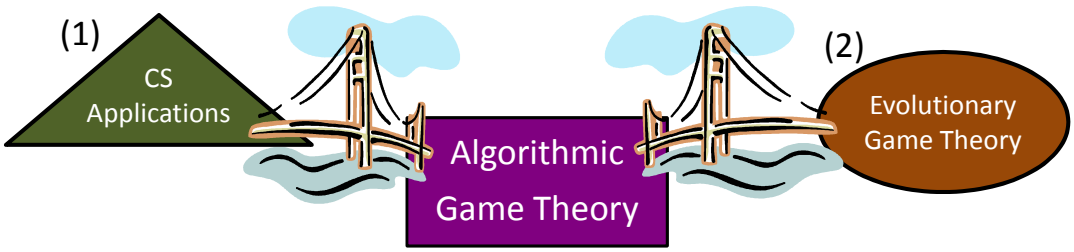


Figure 1: This goals of this work can be viewed as bridge building. The first goal corresponds to the bridge between applied computer science and algorithmic game theory, and the second goal corresponds to the bridge between evolutionary game theory's adaptive play model and algorithmic game theory.

- We consider the evolutionary game theory concept of stochastic stability, and propose the *price of stochastic anarchy* as an alternative to the *price of anarchy* for quantifying the cost of having no central authority. Unlike Nash equilibria, stochastically stable states are the result of natural dynamics of large populations of computationally bounded agents, and are resilient to small perturbations from ideal play. To illustrate the utility of stochastic stability, we study the load balancing game on related machines, which has an unbounded price of anarchy, even in the case of two jobs and two machines. We show that in contrast, in the two player case, the price of stochastic anarchy is 2, and that even in the general case, the price of stochastic anarchy is bounded.

- We propose auction-based mechanisms for admission control of continuous queries to a Data Stream Management System. When submitting a query, each user also submits a bid: how much she is willing to pay for her query to run. The mechanism then determines which queries to admit, and how much to charge each user. Our mechanisms must admit queries and set payments in a way that maximizes system revenue while incentivizing customers to use the system honestly, in the following sense. For the system to be manipulation-resistant, we require that each user maximizes her payoff by bidding her true value of having her query run, and we also propose the notion of sybil-immunity,

that is, that no user can increase her payoff by forging additional false identities. Towards this, we propose several manipulation-resistant payment mechanisms and prove that one guarantees a profit close to a standard profit benchmark, and the others perform well experimentally.

- We study the long standing problem of congestion control at bottleneck routers on the internet. We study how effective some common router queuing policies are when each network endpoint is a self-interested player with no information about the other players' actions or preferences. By employing the adaptive learning model of evolutionary game theory, we study policies such as Droptail, RED, and a policy recently proposed by Gao et al. [46]. We find that while bottleneck routers face heavy congestion at stochastically stable states under Droptail and RED, which are currently widely deployed, the policy of Gao et al. yields fair and efficient conditions with no congestion.

We now begin with a brief summary of the story arcs in algorithmic game theory research that bring us to this juncture. Though computer science initially found its game theory niche in exploring the complexity of computing central solution concepts like Nash equilibria, we present here two other outgrowths of the computer science-game theory merger that have come to form what some consider the "most active" areas of research in algorithmic game theory today [77]: Quantifying the Inefficiency of Equilibria and Algorithmic Mechanism Design. (For an extensive survey of the work that falls under the umbrella of algorithmic game theory, please see [68].)

1.1 QUANTIFYING INEFFICIENCY OF EQUILIBRIA

The primary solution concept in game theory is the well-known Nash equilibrium. In a game, a *Nash equilibrium* is a state where all players have chosen strategies such that, given the other players' current strategy choices, no player can benefit by choosing an alternate strategy. It can be thought of as a "stable state," one where all players feel satisfied in some sense with their choices, or do not look back with regret on their own actions after the game has been played. However, from the perspective of say, the social good, or some

centralized goal, Nash equilibrium can often leave a lot to be desired. There is an expansive body of work on quantifying this "inefficiency" of Nash equilibria. Here, we limit ourselves to a few highlights, to provide some context for our contribution. For a complete survey of the research in this area, we defer to Part Three of the text by Nisan et al [68].

In 1999, Koutsoupias and Papadimitriou [57] initiated the study of "worst-case equilibria." They proposed that analogous to worst-case approximation and worst-case competitiveness, we should study worst-case lack-of-coordination, or "anarchy." Specifically, they studied a load balancing problem and considered the goal of minimizing makespan (the time when the final job is completed). They sought to determine how much worse the makespan is when, instead of having a centralized job scheduling algorithm, the jobs each belong to a player who is able to selfishly choose which server to run his job on. They assumed that rational, self-interested players would end up at a Nash equilibrium solution. The ratio of the value at a worst-case Nash equilibrium to the value at the optimal solution of a game became known as the *price of anarchy*.

Following their paper, a spate of price of anarchy studies followed. Some more notable results include studies on other load balancing variants [24, 62], nonatomic selfish routing (where each player wishes to route a negligible fraction of the overall traffic) [74], atomic selfish routing (where each player routes a nonnegligible fraction of the overall traffic) [44, 9, 22], congestion games (where players vie for resources, say paths in a network, and the cost of using each resource is function of the number of players using it) [81, 20], facility location (where each player wishes to locate her facility to service customers in a way that maximizes her profit) [85], and various network creation/design/formation games [7, 18, 4, 30, 5, 32, 23]. Alongside this body of work, there is also a significant amount of work that studies the price of stability in games. The *price of stability*, initially proposed by [7], is the ratio of the value of the *best* Nash equilibrium to the value of the optimal solution. It can be thought of as an optimist's version of the pessimistic price of anarchy.

However, one downside of the Nash equilibrium solution concept is that while mixed Nash equilibria (Nash equilibria in which players may randomize over their possible strategies) always exist in any game, pure Nash equilibria do not always exist. The notion that players might randomize over many strategies rather than sticking to one strategy is not realistic

or appropriate for many settings. Another drawback is that while Nash equilibria are game states that are resilient to unilateral defection in play, they fail to account for the possibility that two or more players may cooperate and jointly wish defect from a Nash equilibrium state. Due to these limiting properties of Nash equilibria, alternate solution concepts have also been proposed and studied, including: the price of sinking (sink-equilibria always exist in a game and do not refer to a particular stable state like Nash equilibria, but a whole neighborhood of states that a game may indefinitely shuffle between) [48]; approximate equilibria (states where players may benefit from unilateral defection, but not by very much) [8, 18, 4]; and the *strong* price of anarchy (*strong Nash equilibria* are resilient to simultaneous defections by any number of players, making them resilient to collusion among any subset of players) [6, 38, 30, 4].

1.1.1 Our Contribution: the Price of Stochastic Anarchy

A major limitation of the Nash equilibrium solution concept is that while it's clear that once players arrive at such an equilibrium, it is stable (in the sense that players don't benefit from unilaterally changing their strategy), the Nash equilibrium itself comes with no roadmap for how or if players might arrive at such a solution to begin with. And yet another nagging property of Nash equilibria: solutions that seem very unnatural can satisfy the definition of Nash equilibrium. For example consider the following load balancing game.

Example 1.1.1. We are given two machines, M_1 and M_2, and two players, each with a job to run, each with the goal of having their job completed as quickly as possible. Job 1 costs ϵ time to run on M_1, and job 2 costs ϵ time to run on M_2. However, job 1 costs 1 full time unit to run on M_2 and job 2 costs 1 full time unit to run on M_1. (See Figure 2.) Assume that if two jobs are on the same machine, they both reach completion at time equal to the sum of their individual execution times. (So if both jobs are assigned to M_1, then they both finish at time $1 + \epsilon$.) If our goal is to assign the jobs to machines in a way that minimizes makespan, then the optimal solution is obviously to assign job 1 to M_1 and job 2 to M_2. This is also the solution that would be most preferable to the players, as under this assignment, they each finish at time ϵ. However player 1 choosing M_2 and player 2 choosing M_1 is a Nash equilibrium, since both players have chosen the machine where they finish earliest given the choice of the other player. At this suboptimal Nash equilibrium, the makespan is 1, while at the optimal solution, the makespan was ϵ. In this case, the price of anarchy is $1/\epsilon$, an unbounded value (since ϵ can be arbitrarily small)!

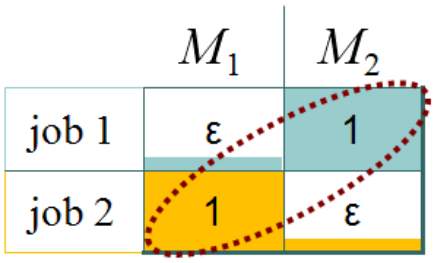


Figure 2: This figure shows the time for running each job on each machine in Example 1.1.1, where Price of Anarchy is unbounded. The assignment of jobs to machines that gives the worst Nash equilibrium is circled.

The optimal solution in this example is a more robust Nash equilibrium in the sense that it is clearly far better for both players than the bad Nash equilibrium; neither player prefers any other outcome. On the other hand, at the bad Nash equilibrium, each player privately wishes she was on the opposing player's machine; if one of them switched to her "preferred" machine, the other would quickly follow suit. These types of brittle Nash are often what make people question whether Nash equilibrium is the most sensible solution concept to study. In light of these many drawbacks, using the *price of anarchy* to measure the cost of the lack of having centralized control may sometimes be too pessimistic.

Motivated by this need for a more robust measure of the price of "anarchy," in Chapter 2 we introduce the notion of the *price of stochastic anarchy*. Stochastic stability is an evolutionary game theory notion proposed in 1990 by Foster and Young [43] within a framework they called adaptive learning. Given a game that is played repeatedly, where the players use given natural heuristics (functions of play history) to decide (or "learn") what strategies to play in each round, the stochastically stable states are those solutions that are robust to noisy behavior (mistakes, or sporadic unexplained behavior by the players) and have positive long-run probability of being played. Such a solution concept by definition allows for computationally bounded, imperfect, players with limited information, and in fact was devised with such players specifically in mind. After finding the stochastically stable states of a

game, there are no remaining questions about how players arrive at such solutions. And all finite games must have stochastically stable states by definition.

In our work, we show that in the load balancing game on unrelated machines, while the price of anarchy is unbounded, the price of stochastic anarchy is bounded. Our work is the first foray into using stochastically stable states as the solution concept when bounding the inefficiency of equilibria. It sows initial seeds for the future development of new analytical techniques needed to obtain such results.

The work in Chapter 2 was completed in collaboration with Katrina Ligett, Kirk Pruhs, and Aaron Roth.

1.2 ALGORITHMIC MECHANISM DESIGN

Algorithmic mechanism design was first defined and proposed by Nisan and Ronen in [67]. They viewed the traditional mechanism design of economists simply as algorithm design where the inputs include strategic players whose actions are determined by their payoff functions. Thus, they proposed the study of mechanism design as applied to "algorithmic problems," like task scheduling, load balancing, or routing, with an emphasis on computational tractability. Following their work, algorithmic mechanism design became a popular area of research. For a complete survey of work in the area, refer to Part Two of the text by Nisan et al [68].

For our purposes we concentrate here on the design of auction mechanisms. An auction mechanism is an algorithm that can be applied any time a set of resources must be allocated to a set of strategic agents. In an auction setting, we assume the agents have private valuations for the resources, and our goal is to assign the resources to the agents and charge them prices in a way that optimizes some objective. Most commonly, the objective is either to maximize *social welfare* (the total valuation of those agents who receive resources), as in the case of the Federal Communications Commission allocating radio spectrum to various telecom companies, or to maximize *profit* (the total payments charged to agents who receive resources), as in the case of Google selling the ads it displays after each user's search query.

The catch is that we also require the algorithm to be designed in such a way so as to make the players always prefer, strategically speaking, to reveal their true valuations when bidding. Such a mechanism is termed *incentive compatible*, *strategy-proof*, or *truthful*. The truthfulness of a mechanism guarantees that if bidders are rational, they will reveal their true valuations, and allow us to determine just how close the system is to reaching its objective (whether it be maximizing social welfare, or maximizing profit). While ensuring truthfulness inevitably leads to sacrifices in profit, guaranteeing truthfulness of a system gives users a sense of security and trust in the system, so even for profit-oriented systems, it can be seen as a long-term investment. This philosophy is apparently held by real-world profit-seekers such as the online auction house, eBay, who uses Vickrey's second-price auction mechanism (a well-known, auction mechanism that only charges bidders the second highest price rather than the winning bid amount in order to guarantee truthfulness).

For our work, we are interested combinatorial auctions. In a combinatorial auction, each bidder has a different valuation for every possible subset of items to be allocated. The type of combinatorial auction of relevance to us is one with single-minded bidders. In such an auction, each bidder desires (has a positive valuation for) a specific subset of items, but has no interest in any other subset. Mu'alem and Nisan [65] studied the case of single-minded bidders, and they characterize the properties of truthful mechanisms for such auctions. Lehmann et al. [60] show that with single-minded bidders, one can find a truthful mechanism that gives a $\sqrt{m}$ approximation to the maximum social welfare, where m is the number of items being sold. In Chapter 3, we propose a single-minded combinatorial auction setting motivated by a recently growing area of databases research. Our goal will be to maximize profit. We now briefly summarize our research in this area.

1.2.1 Our Contribution: Data Streams Query Admission

A Data Stream Management System (DSMS) processes continuous queries about incoming streams of data such as stock market indices, weather data, traffic data, etc. With the growth of internet cloud computing services sold by Amazon, Google, IBM, etc, we envision future DSMSs renting server capacity for their monitoring services to users through the internet.

We propose an auction model for DSMS query admission, in which each user has a private valuation for her query being serviced by the system, and submits a bid that may or may not accurately reflect these valuations. In turn, the system, being of limited processing capacity, chooses which queries to service based on their bids.

The complicating factor in this setting is that the demands of the queries may overlap, so the system capacity required to simultaneously service any two given queries can be less than the sum of the queries' separate individual demands. Given such portions of "shared processing" between queries, optimally choosing which queries to service becomes a great deal more complex. Evidence of this added complexity is that *without* the shared processing or strategic bidders, our resource allocation problem is the classic knapsack problem, in which a subset of items of varying sizes and values must be placed into a limited-capacity knapsack such that the total value of the items chosen is maximized. While there is a polynomial time approximation scheme for the knapsack problem, the densest subgraph problem, which is just a special case of the problem *with* shared processing, has no known polynomial time approximation that is better than a polynomial factor away from optimal[35].

In Chapter 3, we propose four truthful deterministic auction mechanisms that each in some way greedily prioritizes those queries with higher valuations and lower processing demands, and we evaluate their performance experimentally. We propose an additional truthful randomized mechanism that achieves a provable profit guarantee: it always yields at least a profit of $OPT - 2h$, where OPT is the maximum possible profit attainable when there is no requirement of truthfulness, but players who are serviced must all be charged the same price (a price less than their bids), and h is the value of the highest bid. We also consider what happens under these mechanisms when players may cheat by forging additional fake identities, creating dummy queries to manipulate the system. This type of strategic behavior is referred to as a sybil attack. We propose the important property of "sybil immunity," being invulnerable to sybil attack, as desirable design constraint for auction mechanisms. We show that while one of our mechanisms is sybil immune, it does not give a profit guarantee, and the mechanism that gives a profit guarantee is not sybil immune.

The work in Chapter 3 was done in collaboration with Lory Al Moakar, Panos Chrysanthis, Shenoda Guirguis, Alexandros Labrinidis, Panickos Neophytou and Kirk Pruhs.

1.3 CONGESTION AT INTERNET BOTTLENECK ROUTERS

In Chapter 4, we study a long-standing problem from the field of networks: congestion at bottleneck routers on the internet. The internet is a world-wide network whose creation and operation is not controlled by a single, central entity. Instead, separate autonomous entities create and operate their own segments of this world-wide network at will. It has therefore been a long-held belief that the continued dependability and functionality of the internet relies on the sense of civic responsibility and good will of all these separate entities.

In 2002, Akella et al [3] asked the question: *under a simplified model of internet traffic, what would happen if each of the entities, or players, took purely self-interested actions, without regard to other internet users?* They first assumed that network endpoints use the standard Transmission Control Protocol (TCP) for congestion control, which is the most common case for today's internet. Under TCP, an endpoint gradually (additively) increases the number of packets it sends at once, it's "window size," until one of its packets gets dropped due to congestion at a router, at which point it dramatically (multiplicatively) decreases the number of packets it sends at once. They also assumed that routers use the FIFO Droptail queuing policy, also the most common case for today's internet. Under FIFO Droptail, after the router queue fills up, the router simply begins dropping any additional arriving packets. They concluded that under these assumptions, even if endpoints can strategically manipulate their TCP parameters (rates at which they increase and decrease their window sizes), the Nash equilibrium are relatively efficient, in the sense that the router's full capacity is used, and not exceeded by too much. However, they go on to show that under newer congestion control and router policies that are growing in popularity, in which network endpoints are not so responsive to or effected by packet drops, the Nash equilibria are dangerously inefficient, implying possible future "congestion collapse" on the Internet.

Increasing multimedia content on the web has led to an increase in UDP flows on the internet, which, unlike TCP flows, disregard any congestion detected and remain aggressive even in the face of packet drops. Many works have focused on the effect of a single UDP endpoint on other endpoints that are all TCP. Recently, Efraimidis and Tsavlidis [28] proposed a simpler model of endpoints sending traffic through a bottleneck router that we find to be

general enough to capture any type of flow: the one-shot "window game." In the window game, endpoints are not assumed to have specific congestion control properties like TCP or UDP. Instead, each player simply specifies a window size; the number packets they will send at a time, or, to cast it another way, the amount of router capacity they hope to use. In the model, each player's utility, or payoff, is defined as

$$(\text{number of packets that it successfully sends}) - g \cdot (\text{number of its packets that get dropped}).$$

Hence g represents how much each endpoint is hurt by each dropped packet, and endpoints are not assumed specifically to be TCP or UDP. The authors [28] show that in this window game, under Droptail routers, when for example $g \leq 1$, players send more than twice as much traffic as the bottleneck router can handle at the Nash equilibrium.

However, the full-information and mutual-belief-of-rationality assumptions needed for Nash equilibria seem unlikely to hold in a setting as populous and chaotic as the internet. We thus complete the first study of this problem using evolutionary game theory's adaptive learning framework (see Section 1.1.1). Adaptive learning with imitation dynamics is especially suited for this setting because the players need only know what we expect internet endpoints to know: what actions they take and what they themselves experience in each round of play.

Using the window game model, we study the stochastically stable states under the FIFO Droptail and RED router policies and show that they coincide with the unique Nash Equilibrium under both policies. We show that the Nash equilibria and stochastically stable states under these policies are inefficient, with endpoints sending much more traffic than the capacity of the bottleneck router. We then study a policy proposed by Gao et al [46] in which the router drops packets of the greediest flow when the capacity is exceeded, and we show that the only stochastically stable state is efficient and fair: endpoints send no more than bottleneck router capacity, and capacity is divided evenly among them.

The work in Chapter 4 was done in collaboration with Evangelia Pyrga.

2.0 THE PRICE OF STOCHASTIC ANARCHY

2.1 INTRODUCTION

Quantifying the *price of (Nash) anarchy* is one of the major lines of research in algorithmic game theory. Indeed, one fourth of the authoritative algorithmic game theory text edited by Nisan et al. [68] is wholly dedicated to this topic. But the Nash equilibrium solution concept has been widely criticized [48, 14, 33, 34]. First, it is a solution characterization without a road map for how players might arrive at such a solution. Second, at Nash equilibria, players are unrealistically assumed to be perfectly rational, fully informed, and infallible. Third, computing Nash equilibria is PPAD-hard for even 2-player, n-action games [19], and it is therefore considered very unlikely that there exists a polynomial time algorithm to compute a Nash equilibrium even in a centralized manner. Thus, it is unrealistic to assume that selfish agents in general games will converge precisely to the Nash equilibria of the game, or that they will necessarily *converge* to anything at all. In addition, the price of Nash anarchy metric comes with its own weaknesses; it blindly uses the worst case over all Nash equilibria, despite the fact that some equilibria are more resilient than others to perturbations in play.

Considering these drawbacks, computer scientists have paid relatively little attention to if or how Nash equilibria will in fact be reached, and even less to the question of which Nash equilibria are more likely to be played in the event players do converge to Nash equilibria. To address these issues, we employ the stochastic stability framework from evolutionary game theory to study simple dynamics of computationally efficient, imperfect agents. Rather than defining a-priori states such as Nash equilibria, which might not be reachable by natural dynamics, the stochastic stability framework allows us to define a natural dynamic, and

from it derive the stable states. We define the *price of stochastic anarchy* to be the ratio of the worst stochastically stable solution to the optimal solution. The stochastically stable states of a game may, but do not necessarily, contain all Nash equilibria of the game, and so the price of stochastic anarchy may be strictly better than the price of Nash anarchy. In games for which the stochastically stable states are a subset of the Nash equilibria, studying the ratio of the worst stochastically stable state to the optimal state can be viewed as a smoothed analysis of the price of anarchy, distinguishing Nash equilibria that are brittle to small perturbations in perfect play from those that are resilient to noise. (Note that conversely, there may also be stochastically stable states that are not Nash equilibria.)

The evolutionary game theory literature on *stochastic stability* studies n-player games that are played repeatedly. In each round, each player observes her action and its outcome, and then uses simple rules to select her action for the next round based only on her size-restricted memory of the past rounds. In any round, players have a small probability of deviating from their prescribed decision rules. The state of the game is the contents of the memories of all the players. The *stochastically stable states* in such a game are the states with non-zero probability in the limit of this random process, as the probability of error approaches zero. The play dynamics we employ in our work are the imitation dynamics studied by Josephson and Matros [54]. Under these dynamics, each player imitates the strategy that was most successful for her in recent memory.

2.1.1 Our Results

To illustrate the utility of stochastic stability, we study the price of stochastic anarchy of the unrelated load balancing game [10, 6, 38]. To our knowledge, we are the first to quantify the loss of efficiency in any system when the players are in stochastically stable equilibria. In the load balancing game on unrelated machines, even with only two players and two machines, there are Nash equilibria with arbitrarily high cost relative to optimum, and so the price of Nash anarchy is unbounded. We show that these equilibria are inherently brittle, and that for two players and two machines, the price of stochastic anarchy is 2. This result matches the strong price of anarchy [6] without requiring coordination (at strong Nash equilibria, players

have the ability to coordinate by forming coalitions). We further show that in the general n-player, m-machine game, the price of stochastic anarchy is bounded. More precisely the price of stochastic anarchy is upper bounded by the nmth n-step Fibonacci number. We also show that the price of stochastic anarchy is at least $m + 1$.

Our work provides new insight into the equilibria of the load balancing game. Unlike some previous work on dynamics for games, our work does not seek to propose practical dynamics with fast convergence; rather, we use simple dynamics as a tool for understanding the inherent relative stability of equilibria. Instead of relying on player coordination to avoid the Nash equilibria with unbounded cost (as is done in the study of strong equilibria), we show that these bad equilibria are inherently unstable in the face of occasional uncoordinated mistakes. We conjecture that the price of stochastic anarchy is closer to the linear lower bound, paralleling the price of strong anarchy.

In light of our results, we believe the techniques in this work will be useful for understanding the relative stability of Nash equilibria in other games for which the worst equilibria are brittle. Indeed, for a variety of games in the price of anarchy literature, the worst Nash equilibria of the lower bound instances are not stochastically stable.

2.1.2 Related Work

We give a brief survey of related work in three areas: alternatives to Nash equilibria as a solution concept, stochastic stability, and the unrelated load balancing game.

Recently, several authors have noted that the Nash equilibrium is not always a suitable solution concept for computationally bounded agents playing in a repeated game, and have proposed alternatives. Goemans et al. [48] study players who sequentially play myopic best responses, and quantify the *price of sinking* that results from such play. Fabrikant and Papadimitriou [33] propose a model in which agents play restricted finite automata. Blum et al. [14, 13] assume only that players' action histories satisfy a property called *no regret*, and show that for many games, the resulting social costs are no worse than those guaranteed by price of anarchy results.

Although we believe this to be the first work studying stochastic stability in the computer

science literature, computer scientists have recently employed other tools from evolutionary game theory. Fisher and Vöcking [41] show that under replicator dynamics in the routing game studied by Roughgarden and Tardos [74], players converge to Nash. Fisher et al. [40] went on to show that using a simultaneous adaptive sampling method, play converges quickly to a Nash equilibrium. For a thorough survey of algorithmic results that have employed or studied other evolutionary game theory techniques and concepts, see Suri [80].

Stochastic stability and its adaptive learning model as studied in this work were first defined by Foster and Young [43], and differ from the standard game theory solution concept of evolutionarily stable strategies (ESS). ESS are a refinement of Nash equilibria, and so do not always exist, and are not necessarily associated with a natural play dynamic. In contrast, a game always has stochastically stable states that result (by construction) from natural dynamics. In addition, ESS are resilient only to single shocks, whereas stochastically stable states are resilient to persistent noise.

Stochastic stability has been widely studied in the economics literature (see, for example, [87, 55, 58, 15, 29, 73, 54]). We discuss in Section 2.2 concepts from this body of literature that are relevant to our results. We recommend Young [88] for an informative and readable introduction to stochastic stability, its adaptive learning model, and some related results. Our work differs from prior work in stochastic stability in that it is the first to quantify the social utility of stochastically stable states, the *price of stochastic anarchy*.

We also note a connection between the stochastically stable states of the game and the sinks of a game, recently introduced by Goemans et al. as another way of studying the dynamics of computationally bounded agents. In particular, the stochastically stable states of a game under the play dynamics we consider correspond to a subset of the sink equilibria, and so provide a framework for identifying the stable sink equilibria. In potential games, the stochastically stable states of the play dynamics we consider correspond to a subset of the Nash equilibria, thus providing a method for identifying which of these equilibria are stable.

In this work, we study the price of stochastic anarchy in load balancing. Even-Dar et al. [31] show that when playing the load balancing game on unrelated machines, any turn-taking improvement dynamics converge to Nash. Andelman et al. [6] observe that the price of Nash anarchy in this game is unbounded and they show that the strong price of anarchy is linear

in the number of machines. Fiat et al. [38] tighten their upper bound to match their lower bound at a strong price of anarchy of exactly m.

2.2 MODEL AND BACKGROUND

We now formalize (from Young [87]) the adaptive play model and the definition of stochastic stability. We then formalize the play dynamics that we consider. We also provide in this section the results from the stochastic stability literature that we will later use for our results.

2.2.1 Adaptive Play and Stochastic Stability

Let $G = (X, \pi)$ be a game with n players, where $X = \prod_{j=1}^{n} X_i$ represents the strategy sets X_i for each player i, and $\pi = \prod_{j=1}^{n} \pi_i$ represents the payoff functions $\pi_i : X \to \mathbb{R}$ for each player. G is played repeatedly for successive time periods $t = 1, 2, \ldots$, and at each time step t, player i plays some action $s_i^t \in X_i$. The collection of all players' actions at time t defines a play profile $S^t = (S_1^t, S_2^t, \ldots, S_n^t)$. We wish to model computationally efficient agents, and so we imagine that each agent has some finite memory of size z, and that after time step t, all players remember a history consisting of a sequence of play profiles $h^t = (S^{t-z+1}, S^{t-z+2}, \ldots, S^t) \in (X)^z$.

We assume that each player i has some efficiently computable function $p_i : (X)^z \times X_i \to \mathbb{R}$ that, given a particular history, induces a sampleable probability distribution over actions (for all players i and histories h, $\sum_{a \in X_i} p_i(h, a) = 1$). We write p for $\prod_i p_i$. We wish to model imperfect agents who make mistakes, and so we imagine that at time t each player i plays according to p_i with probability $1-\epsilon$, and with probability ϵ plays some action in X_i uniformly at random.[1] That is, for all players i, for all actions $a \in X_i$, $\Pr[s_i^t = a] = (1-\epsilon)p_i(h^t, a) + \frac{\epsilon}{|X_i|}$. The dynamics we have described define a Markov process $P^{G,p,\epsilon}$ with finite state space $H = (X)^z$ corresponding to the finite histories. For notational simplicity, we will write the Markov process as P^ϵ when there is no ambiguity.

[1] The mistake probabilities need not be uniform random—all that we require is that the distribution has support on all actions in X_i.

The potential successors of a history can be obtained by observing a new play profile, and "forgetting" the least recent play profile in the current history.

Definition 2.2.1. For any $S' \in X$, A history $h' = (S^{t-z+2}, S^{t-z+3}, \ldots, S^t, S')$ is a *successor* of history $h^t = (S^{t-z+1}, S^{t-z+2}, \ldots, S^t)$.

The Markov process P^ϵ has transition probability $p^\epsilon_{h,h'}$ of moving from state $h = (S^1, \ldots, S^z)$ to state $h' = (T^1, \ldots, T^z)$:

$$p^\epsilon_{h,h'} = \begin{cases} \prod_{i=1}^n (1-\epsilon)\ p_i(h, T_i^z) + \frac{\epsilon}{|X_i|} & \text{if } h' \text{ is a successor of h;} \\ 0 & \text{otherwise.} \end{cases}$$

We will refer to P^0 as the unperturbed Markov process. Note that for $\epsilon > 0$, $p^\epsilon_{h,h'} > 0$ for every history h and successor h', and that for any two histories h and $\hat{h}$ not necessarily a successor of h, there is a series of z histories $h_1, \ldots, h_z$ such that $h_1 = h$, $h_z = \hat{h}$, and for all $1 < i \leq z$, h_i is a successor of h_{i-1}. Thus there is positive probability of moving between any h and any $\hat{h}$ in z steps, and so P^ϵ is irreducible. Similarly, there is a positive probability of moving between any h and any $\hat{h}$ in $z+1$ steps, and so P^ϵ is aperiodic. Therefore, P^ϵ has a unique stationary distribution μ^ϵ.

The stochastically stable states of a particular game and player dynamics are the states with nonzero probability in the limit of the stationary distribution.

Definition 2.2.2 (Foster and Young [43])**.** A state h is *stochastically stable* relative to P^ϵ if $\lim_{\epsilon \to 0} \mu^\epsilon(h) > 0$.

Intuitively, we should expect a process P^ϵ to spend almost all of its time at its stochastically stable states when ϵ is small.

When a player i plays at random rather than according to p_i, we call this a mistake.

Definition 2.2.3 (Young [87])**.** Suppose $h' = (S^{t-z+1}, \ldots, S^t)$ is a successor of h. A *mistake* in the transition between h and h' is any element S_i^t such that $p_i(h, S_i^t) = 0$. Note that mistakes occur with probability $\leq \epsilon$.

We can characterize the number of mistakes required to get from one history to another.

Definition 2.2.4 (Young [87])**.** For any two states h, h', the *resistance* $r(h, h')$ is the minimum total number of mistakes involved in the transition $h \to h'$ if h' is a successor of h. If h' is not a successor of h, then $r(h, h') = \infty$.

Note that the transitions of zero resistance are exactly those that occur with positive probability in the unperturbed Markov process P^0.

Definition 2.2.5. We refer to the sinks of P^0 as *recurrent classes*. In other words, a *recurrent class* of P^0 is a set of states $C \subseteq H$ such that any state in C is reachable from any other state in C and no state outside C is accessible from any state inside C.

We may view the state space H as the vertex set of a directed graph, with an edge from h to h' if h' is a successor of h, with edge weight $r(h, h')$.

Observation 2.2.6. *We observe that the recurrent classes $H_1, H_2, \ldots$, where each $H_i \subseteq H$, have the following properties:*

1. *From every vertex $h \in H$, there is a path of cost 0 to one of the recurrent classes.*
2. *For each H_i and for every pair of vertices $h, h' \in H_i$, there is a path of cost 0 between h and h'.*
3. *For each H_i, every edge (h, h') with $h \in H_i, h' \notin H_i$ has positive cost.*

Let $r_{i,j}$ denote the cost of the shortest path between H_i and H_j in the graph described above. We now consider the complete directed graph $\mathcal{G}$ with vertex set $\{H_1, H_2, \ldots\}$ in which the edge (H_i, H_j) has weight $r_{i,j}$. Let T_i be a directed minimum-weight spanning in-tree of $\mathcal{G}$ rooted at vertex H_i. (An in-tree is a directed tree where each edge is oriented toward the root.) The *stochastic potential* of H_i is defined to be the sum of the edge weights in T_i.

Young proves the following theorem characterizing stochastically stable states:

Theorem 2.2.7 (Young [87])**.** *In any n-player game G with finite strategy sets and any set of action distributions p, the stochastically stable states of $P^{G,p,\epsilon}$ are the recurrent classes of minimum stochastic potential.*

2.2.2 Imitation Dynamics

In this work, we study agents who behave according to a slight modification of the imitation dynamics introduced by Josephson and Matros [54]. (We note that this modification is of no consequence to the results of Josephson and Matros [54] that we present below.) Player i using imitation dynamics parameterized by $\sigma \in \mathbb{N}$ chooses his action at time $t+1$ according to the following mechanism:

1. Player i selects a set Y of σ play profiles uniformly at random from the z profiles in history h_t.
2. For each play profile $S \in Y$, i recalls the payoff $\pi_i(S)$ he obtained from playing action S_i.
3. Player i plays the action among these that corresponds to his highest payoff; that is, he plays the i^{th} component of $\text{argmax}_{S \in Y} \pi_i(S)$. In the case of ties, he plays a highest-payoff action at random.

The value σ is a parameter of the dynamics that is taken to be $n < \sigma \leq z/2$. These dynamics can be interpreted as modeling a situation in which at each time step, players are chosen at random from a pool of identical players, who each played in a subset of the last z rounds. The players are computationally simple, and so do not counterspeculate the actions of their opponents, instead playing the action that has worked the best for them in recent memory.

We will say that a history h is *monomorphic* if the same action profile S has been repeated for the last z rounds: $h = (S, S, \ldots, S)$. Josephson and Matros [54] prove the following useful fact:

Proposition 2.2.8. *A set of states is a recurrent class of the imitation dynamics if and only if it is a singleton set consisting of a monomorphic state.*

Since the stochastically stable states are a subset of the recurrent classes, we can associate with each stochastically stable state $h = (S, \ldots, S)$ the unique action profile S it contains. This allows us to now define the price of stochastic anarchy with respect to imitation dynamics. For brevity, we will refer to this as simply the price of stochastic anarchy.

Definition 2.2.9. Given a game $G = (X, \pi)$ with a social cost function $\gamma : X \to \mathbb{R}$, the

price of stochastic anarchy of G is equal to $\max \frac{\gamma(S)}{\gamma(\mathbf{OPT})}$, where **OPT** is the play profile that minimizes γ and the max is taken over all play profiles S such that $h = (S, \ldots, S)$ is stochastically stable.

Given a game G, we define the *better response graph* of G: The set of vertices corresponds to the set of action profiles of G, and there is an edge between two action profiles S and S' if and only if there exists a player i such that S' differs from S only in player i's action, and player i does not decrease his utility by unilaterally deviating from S_i to S'_i. Josephson and Matros [54] prove the following relationship between this better response graph and the stochastically stable states of a game:

Theorem 2.2.10. *If $\mathbb{V}$ is the set of stochastically stable states under imitation dynamics, then $V = \{S : (S, \ldots, S) \in \mathbb{V}\}$ is either a strongly connected component of the better response graph of G, or a union of strongly connected components.*

Goemans et al. [48] introduce the notion of sink equilibria and a corresponding notion of the "price of sinking", which is the ratio of the social welfare of the worst sink equilibrium to that of the social optimum. We note that the strongly connected components of the better response graph of G correspond to the sink equilibria (under sequential better-response play) of G, and so Theorem 2.2.10 implies that the stochastically stable states under imitation dynamics correspond to a subset of the sinks of the better response graph of G, and we get the following corollary:

Corollary 2.2.11. *The price of stochastic anarchy of a game G under imitation dynamics is at most the price of sinking of G.*

2.3 LOAD BALANCING: GAME DEFINITION AND PRICE OF NASH ANARCHY

The load balancing game on unrelated machines models a set of agents who wish to schedule computing jobs on a set of machines. The machines have different strengths and weaknesses (for example, they may have different types of processors or differing amounts of memory),

and so each job will take a different amount of time to run on each machine. Jobs on a single machine are executed in parallel such that all jobs on any given machine finish at the same time. Thus, each agent who schedules his job on machine M_i endures the *load* on machine M_i, where the load is defined to be the sum of the running times of all jobs scheduled on M_i. Agents wish to minimize the completion time for their jobs, and social cost is defined to be the *makespan*: the maximum load on any machine.

Formally, an instance of the load balancing game on unrelated machines is defined by a set of n players and m machines $M = \{M_1, \ldots, M_m\}$. The action space for each player is $X_i = M$. Each player i has some cost $c_{i,j}$ on machine j. Denote the cost of machine M_j for action profile S by $C_j(S) = \sum_{i \text{ s.t. } S_i=j} c_{i,j}$. Each player i has utility function $\pi_i(S) = -C_{s_i}(S)$. The social cost of an action profile S is $\gamma(S) = \max_{j \in M} C_j(S)$. We define **OPT** to be the action profile that minimizes social cost: $\mathbf{OPT} = \operatorname{argmin}_{S \in X} \gamma(S)$. Without loss of generality, we will always normalize so that $\gamma(\mathbf{OPT}) = 1$.

The coordination ratio of a game (also known as the price of anarchy) was introduced by Koutsoupias and Papadimitriou [57], and is intended to quantify the loss of efficiency due to selfishness and the lack of coordination among rational agents. Given a game G and a social cost function γ, it is simple to quantify the **OPT** game state S: $\mathbf{OPT} = \operatorname{argmin} \gamma(S)$. It is less clear how to model rational selfish agents. In most prior work it has been assumed that selfish agents play according to a Nash equilibrium, and the price of anarchy has been defined as the ratio of the cost of the worst (pure strategy) Nash state to **OPT**. In this chapter, we refer to this measure as the price of Nash anarchy, to distinguish it from the price of stochastic anarchy, which we defined in Section 2.2.2.

Definition 2.3.1. For a game G with a set of Nash equilibrium states $\mathcal{E}$, the *price of (Nash) anarchy* is $\max_{S \in \mathcal{E}} \frac{\gamma(S)}{\gamma(\mathbf{OPT})}$.

We show here that even with only two players and two machines, the load balancing game on unrelated machines has a price of Nash anarchy that is unbounded by any function of m and n. Consider the two-player, two-machine game with $c_{1,1} = c_{2,2} = 1$ and $c_{1,2} = c_{2,1} = 1/\delta$, for some $0 < \delta < 1$. Then the play profile $\mathbf{OPT} = (M_1, M_2)$ is a Nash equilibrium with cost 1. However, observe that the profile $S^* = (M_2, M_1)$ is also a Nash equilibrium, with cost

$1/\delta$ (since by deviating, players can only increase their cost from $1/\delta$ to $1/\delta + 1$). The price of anarchy of the load balancing game is therefore $1/\delta$, which can be unboundedly large, although $m = n = 2$.

2.4 UPPER BOUND ON PRICE OF STOCHASTIC ANARCHY

The load balancing game is an ordinal potential game [31], and so the sinks of the better-response graph correspond to the pure strategy Nash equilibria. We therefore have by Corollary 2.2.11 that the stochastically stable states are a subset of the pure strategy Nash equilibria of the game, and the price of stochastic anarchy is at most the price of anarchy. We have noted that even in the two-person, two-machine load balancing game, the price of anarchy is unbounded (even for pure strategy equilibria). Therefore, as a warmup, we bound the price of stochastic anarchy of the two-player, two-machine case.

2.4.1 Two Players, Two Machines

Theorem 2.4.1. *In the two-player, two-machine load balancing game on unrelated machines, the price of stochastic anarchy is 2.*

Note that the two-player, two-machine load balancing game can have at most two strict pure strategy Nash equilibria. (For brevity we consider the case of strict equilibria. The argument for weak equilibria is similar). Note also that either there is a unique Nash equilibrium at (M_1, M_1) or (M_2, M_2), or there are two at $N_1 = (M_1, M_2)$ and $N_2 = (M_2, M_1)$.

An action profile N *Pareto dominates* N' if for each player i, $C_{N_i}(N) \leq C_{N'_i}(N')$.

Lemma 2.4.2. *If there are two Nash equilibria, and N_1 Pareto dominates N_2, then only N_1 is stochastically stable (and vice versa).*

Proof. Note that if N_1 Pareto dominates N_2, then it also Pareto dominates (M_1, M_1) and (M_2, M_2), since each is a unilateral deviation from a Nash equilibrium for both players. Consider the monomorphic state $(N_2, \ldots, N_2)$. If both players make simultaneous mistakes at time t to N_1, then by assumption, N_1 will be the action profile in $h_{t+1} = (N_2, \ldots, N_2, N_1)$

with lowest cost for both players. Therefore, with positive probability, both players will draw samples of their histories containing the action profile N_1, and therefore play it, until $h_{t+z} = (N_1, \ldots, N_1)$. Therefore, there is an edge in $\mathcal{G}$ from $h = \{N_2, \ldots, N_2\}$ to $h' = \{N_1, \ldots, N_1\}$ of resistance 2. However, there is no edge from h' to any other state in $\mathcal{G}$ with resistance $< \sigma$. Recall our initial observation that in fact, N_1 Pareto dominates all other action profiles. Therefore, no set of mistakes will yield an action profile with higher payoff than N_1 for either player, and so to leave state h' will require at least σ mistakes (so that some player may draw a sample from their history that contains no instance of action profile N_1). Therefore, given any minimum spanning in-tree of $\mathcal{G}$ rooted at h, we may add an edge (h, h') of weight 2, and remove the outgoing edge from h', which we have shown must have cost $\geq \sigma$. This is a minimum spanning tree rooted at h' with strictly lower cost. We have therefore shown that h' has strictly lower stochastic potential than h, and so by Theorem 2.2.7, h is not stochastically stable. Since at least one Nash equilibrium must be stochastically stable, $h' = (N_1, \ldots, N_1)$ is the unique stochastically stable state. □

Proof of Theorem 2.4.1. If there is only one Nash equilibrium (M_1, M_1) or (M_2, M_2), then it must be the only stochastically stable state (since in potential games these are a nonempty subset of the pure strategy Nash equilibria), and must also be **OPT**. In this case, the price of anarchy is equal to the price of stochastic anarchy, and is 1. Therefore, we may assume that there are two Nash equilibria, N_1 and N_2. If N_1 Pareto dominates N_2, then N_1 must be **OPT** (since load balancing is a potential game), and by Lemma 2.4.2, N_1 is the only stochastically stable state. In this case, the price of stochastic anarchy is 1 (strictly less than the (possibly unbounded) price of anarchy). A similar argument holds if N_2 Pareto dominates N_1. Therefore, we may assume that neither N_1 nor N_2 Pareto dominate the other.

Without loss of generality, assume that N_1 is **OPT**, and that in $N_1 = (M_1, M_2)$, M_2 is the maximally loaded machine. Suppose that M_2 is also the maximally loaded machine in N_2. (The other case is similar.) Together with the fact that N_1 does not Pareto dominate

N_2, this gives us the following:

$$\begin{aligned} c_{1,1} &\leq c_{2,2} \\ c_{2,1} &\leq c_{2,2} \\ c_{1,2} &\geq c_{2,2} \end{aligned}$$

From the fact that both N_1 and N_2 are Nash equilibria, we get:

$$\begin{aligned} c_{1,1} + c_{2,1} &\geq c_{2,2} \\ c_{1,1} + c_{2,1} &\geq c_{1,2} \end{aligned}$$

In this case, the price of anarchy among pure strategy Nash equilibria is:

$$\frac{c_{1,2}}{c_{2,2}} \leq \frac{c_{1,1} + c_{2,1}}{c_{2,2}} \leq \frac{c_{1,1} + c_{2,1}}{c_{1,1}} = 1 + \frac{c_{2,1}}{c_{1,1}}$$

Similarly, we have:

$$\frac{c_{1,2}}{c_{2,2}} \leq \frac{c_{1,1} + c_{2,1}}{c_{2,2}} \leq \frac{c_{1,1} + c_{2,1}}{c_{2,1}} = 1 + \frac{c_{1,1}}{c_{2,1}}$$

Combining these two inequalities, we get that the price of Nash anarchy is at most $1 + \min(c_{1,1}/c_{2,1}, c_{2,1}/c_{1,1}) \leq 2$. Since the price of stochastic anarchy is at most the price of anarchy over pure strategies, this completes the proof. □

2.4.2 General Case: n Players, m Machines

Theorem 2.4.3. *The general load balancing game on unrelated machines has price of stochastic anarchy bounded by a function Ψ depending only on n and m, and*

$$\Psi(n, m) \leq m \cdot F_{(n)}(nm + 1),$$

where $F_{(n)}(i)$ denotes the i^{th} n-step Fibonacci number.[2]

To prove this upper bound, we show that any solution worse than our upper bound cannot be stochastically stable. To show this impossibility, we take any arbitrary solution worse than our upper bound and show that there must always be a minimum cost in-tree in $\mathcal{G}$ rooted at a different solution that has strictly less cost than the minimum cost in-tree rooted at that solution. We then apply Proposition 2.2.8 and Theorem 2.2.7. The proof proceeds by a series of lemmas.

Definition 2.4.4. For any monomorphic Nash state $h = (S, \ldots, S)$, let the *Nash Graph* of h be a directed graph with vertex set M and directed edges (M_i, M_j) if there is some player i with $S_i = M_i$ and $\mathbf{OPT}_i = M_j$. Let the *closure* $\bar{M}_i$ of machine M_i, be the set of states reachable from M_i by following 0 or more edges of the Nash graph.

Lemma 2.4.5. *In any monomorphic Nash state $h = (S, \ldots, S)$, if there is a machine M_i such that $C_i(S) > m$, then every machine $M_j \in \bar{M}_i$ has cost $C_j(S) > 1$.*

Proof. Suppose this were not the case, and there exists an $M_j \in \bar{M}_i$ with $C_j(S) \leq 1$. Since $M_j \in \bar{M}_i$, there exists a simple path $(M_i = M_1, M_2, \ldots, M_k = M_j)$ with $k \leq m$. Since S is a Nash equilibrium, it must be the case that $C_{k-1}(S) \leq 2$ because by the definition of the Nash graph, the directed edge from M_{k-1} to M_k implies that there is some player i with $S_i = M_{k-1}$, but $\mathbf{OPT}_i = M_k$. Since $1 = \gamma(\mathbf{OPT}) \geq C_k(\mathbf{OPT}) \geq c_{i,k}$, if player i deviated from his action in Nash profile S to $S'_i = M_k$, he would experience cost $C_k(S) + c_{i,k} \leq 1 + 1 = 2$. Since he cannot benefit from deviating (by definition of Nash), it must be that his cost in S, $C_{k-1}(S) \leq 2$. By the same argument, it must be that $C_{k-2}(S) \leq 3$, and by induction, $C_1(S) \leq k \leq m$. □

[2] $F_{(n)}(i) = \begin{cases} 1 & \text{if } i \leq n; \\ \sum_{j=i-n}^{i} F_{(n)}(j) & \text{otherwise.} \end{cases}$ $F_{(n)}(i) \in o(2^i)$ for any fixed n.

Lemma 2.4.6. *For any monomorphic Nash state $h = (S, \ldots, S) \in \mathcal{G}$ with $\gamma(S) > m$, there is an edge from h to some $g = (T, \ldots, T)$ where $\gamma(T) \leq m$ with edge cost $\leq n$ in $\mathcal{G}$.*

Proof. Let $D = \{M_j : C_i(S) \geq m\}$, and define the closure of D, $\bar{D} = \bigcup_{M_i \in D} \bar{M}_i$. Consider the successor state h' of h that results when every player i such that $S_i^t \in \bar{D}$ makes a mistake and plays on their OPT machine $S_i^{t+1} = \mathbf{OPT}_i$, and all other players do not make a mistake and continue to play $S_i^{t+1} = S_i^t$. Note that by the definition of $\bar{D}$, for $M_j \in \bar{D}$, for all players i playing machine j in S, $\mathbf{OPT}_i \in \bar{D}$. Let $T = S^{t+1}$. Then for all j such that $M_j \in \bar{D}$, $C_j(T) \leq 1$, since $C_j(T) \leq C_j(\mathbf{OPT}) \leq 1$. To see this, note that for every player i such that $S_i^t = M_j \in \bar{D}$, $S_i^{t+1} = M_j$ if and only if $\mathbf{OPT}_i = M_j$. Similarly, for every player i such that $S_i^{t+1} = M_j \in \bar{D}$ but $S_i^t \neq M_j$, $\mathbf{OPT}_i = M_j$, and so for each machine $M_j \in \bar{D}$, the agents playing on M_j in T are a subset of those playing on M_j at $\mathbf{OPT}$. Note that by Lemma 2.4.5, for all $M_j \in \bar{D}$, $C_j(S) > 1$. Therefore, for every agent i with $S_i^t \in \bar{D}$, $\pi_i(T) > \pi_i(S)$, and so for $h'' = (S, \ldots, S, T, T)$ a successor of h', $r(h', h'') = 0$. Reasoning in this way, there is a path of zero resistance from h' to $g = (T, \ldots, T)$. We have therefore exhibited a path between h and g that involves only $|\{i : S_i^t \in \bar{D}\}| \leq n$ mistakes. Finally, we observe that if $M_j \in \bar{D}$ then $C_j(T) \leq 1$, and by construction, if $M_j \notin \bar{D}$, then $C_j(T) = C_j(S) < m$, since as noted above $M_j \notin \bar{D}$ implies that the players playing M_j in S are the same set playing M_j in T. Thus, we have $\gamma(T) \leq m$, which completes the proof. □

Lemma 2.4.7. *Let $h = (S, \ldots, S) \in \mathcal{G}$ be any monomorphic state with $\gamma(S) \leq m$. Any path in $\mathcal{G}$ from h to a monomorphic state $h' = (S', \ldots, S') \in \mathcal{G}$ where $\gamma(h') > m \cdot F_{(n)}(mn + 1)$ must contain an edge with cost $\geq \sigma$, where $F_{(n)}(i)$ denotes the i^{th} n-step Fibonacci number.*

Proof. Suppose there were some directed path $\mathcal{P}$ in $\mathcal{G}$ ($h = h_1, h_2, \ldots, h_l = h'$) such that all edge costs were less than σ. We will imagine assigning costs to players on machines adversarially: for a player i on machine M_j, we will consider $c_{i,j}$ to be undefined until play reaches a monomorphic state h_k in which he occupies machine j, at which point we will assign $c_{i,j}$ to be the highest value consistent with his path from h_{k-1} to h_k. Note that since initially $\gamma(S) \leq m$, we must have for all $i \in N$, $c_{i,S_i} \leq m = mF_{(n)}(n)$.

There are mn costs $c_{i,j}$ that we may assign, and we have observed that our first n assignments have taken values $\leq mF_{(n)}(n) = mF_{(n)}(1)$. We will assume inductively that our

k^{th} assignment takes value at most $mF_{(n)}(k)$. Let $h_k = (T, \ldots, T)$ be the last monomorphic state in $\mathcal{P}$ such that only k cost assignments have been made, and $h_{k+1} = (T', \ldots, T')$ be the monomorphic state at which the $k+1^{st}$ cost assignment is made for some player i on machine M_j. Since by assumption, fewer than σ mistakes are made in the transition $h_k \to h_{k+1}$, it must be that $c_{i,j} \leq C_{T_i}(T)$; that is, $c_{i,j}$ can be no more than player i's experienced cost in state T. If this were not so, player i would not have continued playing on machine j in T' without additional mistakes, since with fewer than σ mistakes, any sample of size σ would have contained an instance of T which would have yielded higher payoff than playing on machine j. Note however that the cost of any machine M_j in T is at most:

$$C_j(T) \leq \sum_{i: c_{i,j} \neq \text{ undefined}} c_{i,j} \leq \sum_{i=0}^{n-1} mF_{(n)}(k-i) = mF_{(n)}(k+1)$$

where the inequality follows by our inductive assumption. We have therefore shown that the k^{th} cost assigned is at most $mF_{(n)}(k)$, and so the claim follows since there are at most nm costs $c_{i,j}$ that may be assigned, and the cost on any machine in S' is at most the sum of the n highest costs. □

of Theorem 2.4.3. Given any state $h = (S, \ldots, S) \in \mathcal{G}$ where $\gamma(S) > m \cdot F_{(n)}(mn+1)$, we can exhibit a state $f = (U, U, \ldots, U)$ with lower stochastic potential than h such that $\gamma(U) \leq m \cdot F_{(n)}(nm+1)$ as follows.

Consider the minimum weight spanning in-tree T_h of $\mathcal{G}$ rooted at h. We will use it to construct a spanning in-tree T_f rooted at a state f as follows: We add an edge of cost at most n from h to some state $g = (T, \ldots, T)$ such that $\gamma(T) \leq m$ (such an edge is guaranteed to exist by Lemma 2.4.6). This induces a cycle through h and g. To correct this, we remove an edge on the path from g to h in T_h of cost $\geq \sigma$ (such an edge is guaranteed to exist by Lemma 2.4.7). Since this breaks the newly induced cycle, we now have a spanning in-tree T_f with root $f = (U, U, \ldots, U)$ such that $\gamma(U) \leq m \cdot F_{(n)}(mn+1)$. Since the added edge has lower cost than the removed edge, T_f has lower cost than T_h, and so f has lower stochastic potential than h.

Since the stochastically stable states are those with minimum stochastic potential by Theorem 2.2.7 and Proposition 2.2.8, we have proven that h is not stochastically stable. □

	M_1	M_2	M_3	M_4
1	1	$1-\delta$	∞	∞
2	$2-2\delta$	1	$2-3\delta$	∞
3	$3-4\delta$	∞	1	$3-5\delta$
4	$4-6\delta$	∞	∞	1

Figure 3: A load-balancing game with price of stochastic anarchy m for $m = 4$. The entry corresponding to player i and machine M_j represents the cost $c_{i,j}$. The δs represent some sufficiently small positive value and the ∞s can be any sufficiently large value. The optimal solution is (M_1, M_2, M_3, M_4) and costs 1, but (M_2, M_3, M_4, M_1) is also stochastically stable and costs $4 - 6\delta$. This example can be easily generalized to arbitrary m.

2.5 LOWER BOUND ON PRICE OF STOCHASTIC ANARCHY

In this section, we show that the price of stochastic anarchy for load balancing is at least m, the price of strong anarchy. We show this by exhibiting an instance for which the worst stochastically stable solution costs m times the optimal solution. Our proof that this bad solution is stochastically stable uses the following lemma to show that the min cost in-tree rooted at that solution in $\mathcal{G}$ has cost as low as the min cost in-tree rooted at any other solution. We then simply apply Theorem 2.2.7 and Proposition 2.2.8.

Lemma 2.5.1. *For two monomorphic states h and h' corresponding to play profiles S and S', if S' is a unilateral better response deviation from S by some player i, then the resistance $r(h, h') = 1$.*

Proof. Suppose player i makes the mistake of playing S_i' instead of S_i. Since this is a better-response move, he experiences lower cost, and so long as he samples an instance of S', he will continue to play S_i'. No other player will deviate without a mistake, and so play will reach monomorphic state h' after z turns. □

Theorem 2.5.2. *The price of stochastic anarchy of the load balancing game on unrelated machines is at least m.*

Proof. To aid in the illustration of this proof, refer to the instance of the load balancing game pictured in Fig. 3. Consider the instance of the load balancing game on m unrelated machines where $n = m$ and the costs are as follows. For each player i from 1 to n, let $c_{i,i} = 1$. For each player i from 2 to n, let $c_{i,1} = i - 2(i-1)\delta$, where δ is a diminishingly small positive integer. Finally, for each player i from 1 to $n-1$, let $c_{i,i+1} = i - (2i-1)\delta$. Let all other costs be ∞ or some sufficiently large positive value.

Note that in this instance the optimal solution is achieved when each player i plays on machine M_i and thus $\gamma(\mathbf{OPT}) = 1$. Also note that the only pure-strategy Nash states in this instance are the profiles

$$\begin{aligned}
N_1 &= (M_1, M_2, \ldots, M_m), \\
N_2 &= (M_2, M_1, M_3, M_4, \ldots, M_m), \\
N_3 &= (M_2, M_3, M_1, M_4, \ldots, M_m), \\
&\ldots \\
N_{m-1} &= (M_2, M_3, M_4, \ldots, M_{m-1}, M_1, M_m), \\
N_m &= (M_2, M_3, M_4, \ldots, M_m, M_1).
\end{aligned}$$

We observe that $\gamma(N_m) = m - 2(m-1)\delta \approx m$, and the monomorphic state corresponding to N_m is stochastically stable:

Note that for the monomorphic state corresponding to each Nash profile N_i, there is an edge of resistance 2 to any monomorphic state $(S_i, \ldots, S_i)$ where S_i is on a better-response path to Nash profile N_{i+1}. This transition can occur with two simultaneous mistakes as follows: At the same time step t, player i plays on machine M_{i+1}, and player $i+1$ plays on machine M_i. Since for this turn, player i plays on machine M_{i+1} alone, he experiences cost that is δ less than his best previous cost. Player $i+1$ experiences higher cost. Therefore, player $i+1$ returns to machine M_{i+1} and continues to play it (since N_i continues to be the play profile in his history for which he experienced lowest cost). Player i continues to sample the play profile from time step t for the next σ rounds, and so continues to play on M_{i+1}

without further mistakes (even though player $i+1$ has now returned). In this way, play proceeds in z timesteps to a new monomorphic state S_i without any further mistakes. Note that in S_i, players i and $i+1$ both occupy machine M_{i+1}, and so S_i is one better-response move, and hence one mistake, away from N_{i+1} (by moving to machine M_1, player $i+1$ can experience δ less cost).

Finally, we construct a minimum spanning in-tree T_{N_m} from the graph $\mathcal{G}$ rooted at N_m. For the monomorphic state corresponding to the Nash profile N_i, $1 \leq i \leq m-1$, we include the resistance 2 edge to S_i. All other monomorphic states correspond to non-Nash profiles, and so are on better-response paths to some Nash state (since this is a potential game). When a state is on a better-response path to two Nash states N_i and N_j, we consider only the state N_i such that $i > j$. For each non-Nash monomorphic state, we insert the edge corresponding to the first step in the better-response path to N_i, which by Lemma 2.5.1 has cost 1. Since non-Nash monomorphic states are part of shortest-path in-trees to Nash monomorphic states, which have edges to Nash states of higher index, this process produces no cycles, and so forms a spanning in-tree rooted at N_m. Moreover, no spanning tree of $\mathcal{G}$ can have lower cost, since every edge in T_{N_m} is of minimal cost: the only edges in T_{N_m} that have cost > 1 are those leaving strict Nash states, but *any* edge leaving a strict Nash state must have cost ≥ 2. Therefore, by definition of stochastic potential, Theorem 2.2.7, and Proposition 2.2.8, the monomorphic state corresponding to N_m is stochastically stable. □

Remark 2.5.3. *More complicated examples than the one we provide here show that the price of stochastic anarchy is greater than m, and so our lower bound is not tight. For an example, see Figure 4.*

We note the exponential separation between our upper and lower bounds. We conjecture, however, that the true value of the price of stochastic anarchy falls closer to our lower bound:

Conjecture 2.5.4. *The price of stochastic anarchy in the load balancing game with unrelated machines is $O(m)$.*

If this conjecture is correct, then the $O(m)$ bound from the strong price of anarchy [6] can be achieved without coordination.

	M_1	M_2	M_3	M_4
1	1	1	∞	$4-3\delta$
2	$2-\delta$	1	$2-\delta$	∞
3	$3-2\delta$	$3-2\delta$	1	$3-2\delta$
4	$4-3\delta$	$5-4\delta$	∞	1

Figure 4: The optimal solution here is (M_1, M_2, M_3, M_4) and costs 1, but by similar reasoning as in the proof of Theorem 2.5.2, (M_4, M_3, M_1, M_2) is also stochastically stable and costs $5-4\delta$. This example can be easily generalized to arbitrary values of m.

2.6 SUMMARY

In this chapter of our work, we used the evolutionary game theory solution concept of stochastic stability as a tool for quantifying the inefficiency that results when there is no central authority. As opposed to "price of anarchy," which assumes players play Nash equilibrium solutions, we proposed the "price of stochastic anarchy," which can capture the relative stability of equilibria. When the stochastically stable states of a game are a subset of the Nash equilibria, the price of stochastic anarchy can be viewed as a smoothed analysis of the price of anarchy, quantifying the inefficiency of the worst Nash equilibria that are resilient to small perturbations.

We showed that in the load balancing game on unrelated machines, for which the price of Nash anarchy is unbounded, the "bad" Nash equilibria are not stochastically stable, and so the price of stochastic anarchy is bounded. We conjecture that the upper bound given in this work is not tight and the cost of stochastic stability for load balancing is $O(m)$. If this conjecture is correct, it implies that the fragility of the "bad" equilibria in this game is attributable to their instability, not only in the face of player coordination, but also to minor uncoordinated perturbations in play.

We expect that the techniques used in this work will also be useful in understanding the

relative stability of Nash equilibria in other games for which the worst equilibria are brittle. This promise is evidenced by the fact that the worst Nash in the worst-case instances in many games (for example, the Roughgarden and Tardos [74] lower bound showing an unbounded price of anarchy for routing unsplittable flow) are not stochastically stable.

Another direction of future research is to address the practical question of convergence time to stochastically stable states. While the imitation dynamics studied here most likely do not admit good convergence-time guarantees, it might be possible that different dynamics or a modification of imitation dynamics do allow for fast convergence. One might also study states have positive probability in the "short-run" or "medium-run," (e.g., states that have positive probability within some polynomially-bounded number of rounds), rather than only considering those that are stable in the long-run.

3.0 CONTINUOUS QUERY ADMISSION CONTROL

In the last chapter we completed work toward our goal of demonstrating the utility of evolutionary game theory's adaptive learning model. In this chapter, we discuss work that we completed toward our goal of using game theoretic models for problems in applied computer science. Specifically, we study a problem from the field of data management, and we do so by using a much stronger equilibrium concept: dominant strategy equilibria. We learned in the last chapter that at a Nash equilibrium, each player's strategy is payoff maximizing *given* what the other players are playing. In contrast, at a *dominant strategy equilibrium*, each player's strategy must be payoff maximizing *regardless of what the other players do*.

3.1 INTRODUCTION

The growing need for *monitoring applications* such as the real-time detection of disease outbreaks, tracking the stock market, environmental monitoring via sensor networks, and personalized and customized Web alerts, has led to a paradigm shift in data processing paradigms, from Database Management Systems (DBMSs) to Data Stream Management Systems (DSMSs) (e.g., [1, 52, 75]). In contrast to DBMSs in which data is stored, in DSMSs, monitoring applications register Continuous Queries (CQs) which continuously process unbounded data streams looking for data that represent events of interest to the end-user.

We consider the setting of a business that seeks to profit from selling data stream monitoring/management services. One might imagine that a DSMS rents server capacity to users similar to the way Amazon, Google, and IBM now sell cloud computing services [72, 12, 63]. Auctions, used for example by Google to sell search engine ad words, are a proven way of

both maximizing a system's potential profit, as well as appealing to the end-user. Instead of a business selling their services at a set price, an auction mechanism (soliciting bids, then selecting winners) allows a system to charge prices per user based on what the individual user is willing to pay. Users who don't get serviced are not denied arbitrarily due to the system's limited resources, but instead feel legitimately excluded because their bid was simply not high enough. And perhaps most compellingly, an auction setting allows the system to subtly control the balance between overloading their servers and charging the most profitable prices. Hence, we investigate auction-based mechanisms for admission control of CQs to the DSMS.

One of the key challenges to designing these auction mechanisms is determining how to best take advantage of the potential shared processing between CQs. The fact that some queries can share resources obfuscates each query's actual load on the system. Without clear-cut knowledge of each query's load on the system, optimally selecting the queries to admit becomes exceedingly challenging from a combinatorial perspective. On top of this, we must also address the game theoretic concern of guaranteeing the mechanism we design is not manipulable by users. Specifically, we desire that the mechanism is *strategyproof* (also known as *truthful*), which means a client always maximizes her payoff by bidding her true valuation for having her query run, regardless of how the other players bid.

Our Contributions. Our contributions are as follows (they are described in further detail in Section 3.1.3):

- We apply techniques and principles from algorithmic game theory to a data streams query admission control problem. In doing so, we introduce a new auction problem (that can be posed abstractly and applied quite generally).
- We introduce the notion of *sybil immunity* for auction mechanisms: the desirable property that players cannot benefit by taking on additional false identities in the system. (We note that this concept can be generally applied to any mechanism design setting.)
- We propose greedy and randomized algorithms for this problem and show that they are *strategyproof*, but the greedy approaches do not admit good profit guarantees, and while the randomized approach guarantees a profit amount close to a standard profit benchmark, it is not sybil immune.

- We experimentally show that greedy *profit density*-based mechanisms (which take into account both the bid and the load for each user's query), usually provide both increased system profits as well as better user payoff, compared to the randomized algorithm.

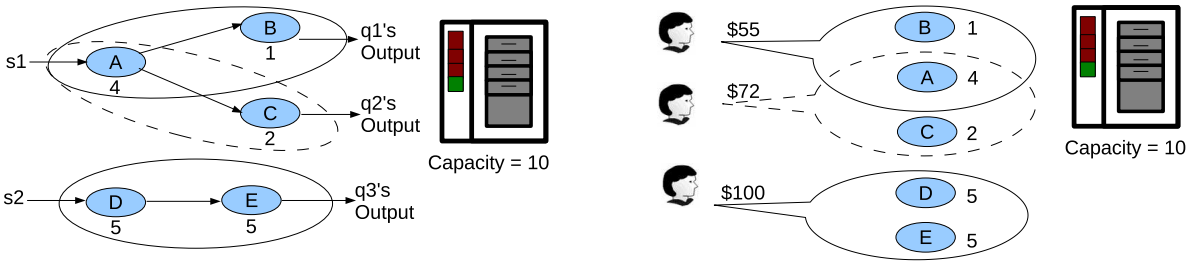
Road map. We continue the introduction by defining the system model and formalizing the problem statement in Section 3.1.1. We then provide a summary of the relevant background from the auction and game theory literature in Section 3.1.2. With these definitions and background, we are ready to give a high-level description of the contributions of this work (in Section 3.1.3). Next, we first present our greedy mechanisms (in Section 3.2), and prove their strategyproofness. We then give a mechanism in Section 3.3 that provably approximates a standard profit benchmark. We analyze the vulnerability and robustness of our mechanisms to sybil attack in Section 3.4. Finally, we present our experimental results in Section 3.5.

3.1.1 System Model and Problem Statement

We assume that each of n users submits a continuous query q_i along with a bid b_i. The bid expresses a declared bound on how much she is willing to pay to execute the query over a fixed time period (say a day for concreteness). Further each user has a private value v_i expressing how much having query q_i run is really worth to her. It will be useful to define a variable h to represent the largest valuation of any user. For our purposes, it is sufficient to view a continuous query as a collection of operators. For example, a continuous query might consist of three operators:

- An operator on a stream of stock quotes that selects out high value transactions,
- An operator on a stream of news stories that selects stories that mention companies with publicly traded stock, and
- A join operator that joins the results of the two selection operators on the company name attribute.

We assume that each operator o_j has an associated load c_j that represents the fraction of the system's capacity that this operator will use, and this load can at least be reasonably approximated by the system (say maybe from historical data). In order for a query q_i to be successfully serviced, all of the operators in q_i must be processed. The *payoff* (aka utility)



(a) The queries as seen by the DSMS. Query q_1 has two operators A and B, query q_2 is formed of A followed by C, and query q_3 is composed of operator D followed by E. As shown, operator A is shared between queries q_1 and q_2. Each operator is also labeled with the load associated with it. Operator A for example incurs a load of 4 units. The input data streams to operators A and D are labeled $s1$ and $s2$ respectively.

(b) The query plan is simplified here to show only the operators without information about the flow of data. This simplified, abstract information model is the one we use for our problem. In our problem, each user submits a bid: how much she claims to value her query being serviced. The bids are shown above as dollar amounts.

Figure 5: Two different views of the sample input of Example 3.1.1.

u_i of the user that submitted query q_i is $v_i - p_i$ if q_i is accepted, and 0 otherwise. Note that many CQs may contain the same operator. For example, one could imagine many queries want to select news stories on publicly traded companies.

To make these concepts concrete, consider Example 3.1.1 (depicted in Figures 5(a) and 5(b)). For the purposes of the problem we are studying, it is sufficient to abstract away the dependencies between operators and retain only the information seen in Figure 5(b): the set of operators that comprise all the queries, the load of each operator, and an indication of which queries each operator is used in. We thus retain knowledge of which operators are shared between which queries. We also take each user's bid as part of our input.

Example 3.1.1. Three queries (q_1, q_2 and q_3) are submitted to a DSMS with a capacity of 10 units. Figure 5(a) shows the query plan of q_1, q_2 and q_3. Note that q_1 and q_2 share operator A. Figure 5(b), which depicts the relevant information model for the problem we study in this work, shows that users 1, 2, and 3 bid \$55 , \$72 and \$100 respectively.

In our model, the DSMS has an admission control mechanism that, at the end of each day, accepts the bids and relevant information about the queries, and returns a decision

about which queries to admit and run the next day. The mechanism also returns the price p_i charged to those admitted. The aggregate load of the operators in the accepted queries can be at most the capacity of the server. The *payoff* (aka utility) u_i of the user that submitted query q_i is $v_i - p_i$ if q_i is accepted, and 0 otherwise.

We assume an underlying query model similar to the Aurora query model [1] where subnetworks are connected via connection points. During the transition phase at the end of each day, the upstream connection points that surround the subnetworks that need to be modified hold any incoming data tuples. The tuples stored inside the queues of these subnetworks are drained through the downstream connection points. Then the query planner modifies the subnetwork by adding new operators or deleting operators. Once the query planner finishes, the tuples stored at the connection points are input into the subnetwork before the newly arriving tuples are executed. This transition phase ensures the correctness of the results output by queries that continue to execute for the next day.

As opposed to our work that focuses on admission control on the registration of CQs, other works on admission control in data streams focus on the arrival of data and in particular shedding data at the source, e.g. [83, 69, 11, 84]. The goal of "load shedding" is to reduce the load on the DSMS by dropping tuples from incoming data streams so that quality of service with respect to latency is maintained for each customer's query (sometimes at the expense of accuracy). We propose that admission control actually begin at the query level; that the first filter of the DSMS should be to selectively choose which queries to register to begin with. Our work, in concert with these load shedding works, can ensure the system resources are not overloaded while being efficiently used and profitably run.

From the business' point of view, the most obvious design goal for the mechanism is to maximize profit, which is the aggregate prices charged to the accepted queries. Another first class design goal for the mechanism is *strategyproofness* (also known as *incentive compatibility* or *truthfulness*). Intuitively, strategyproofness means that users have no incentive to "lie" about their private values, or bid strategically. Auction-based profit-driven businesses like eBay and Google AdWords attempt to design and use strategyproof auction mechanisms, even at the expense of potential short-term profit, because when users perceive that the system is manipulable, they have less trust in the system and are less likely to continue

using it. Hence requiring that their auction mechanisms be strategyproof is an investment in their long-term success.

3.1.2 Relevant Background on Auctions

In many auction settings, bidders' true valuations are the only private information, and hence in such settings *strategyproofness* means that users/bidders maximize their payoff when bidding their true valuations. In this chapter, we will refer to this property as *bid-strategyproofness*, as we will later be considering other private information the users can potentially hide from the system. Several standard auction problems are special cases of our auction problem for continuous queries.

Settings without Sharing. In the special case that there are no shared operators, the load of each query (which is the aggregate load of the query's operators) is the same, and there is room for k queries, then this is equivalent to the problem of auctioning k identical goods. Charging each of the k highest bidders the $(k+1)$st highest bid is well known to be bid-strategyproof. When $k = 1$, this is famously known as "Vickrey's second price" auction.

If CQs do not share operators, but the load of each query may be different, then the resulting problem is what is known in the literature as a Knapsack Auction problem [2]. Aggarwal and Hartline [2] give a bid-strategyproof randomized mechanism that guarantees an expected profit of at least

$$\alpha OPT - \gamma h \log\log\log n.$$

Here α and γ are constants, OPT is the optimal profit that can be obtained from monotone pricing. Monotone pricing means that higher load queries must be charged higher prices. Recall that h is the maximum valuation. Intuitively the additive loss of h in any guarantee of this kind is unavoidable because it is not possible for a bid-strategyproof mechanism to obtain a profit competitive with OPT if there is only one user with high valuation. This is proved more formally in [51].

Operator Sharing. Operators shared between queries greatly complicate the task of the mechanism because the profit *density* of a query, which refers to the ratio of the bid

for that query (or potential profit to be obtained from accepting the query) to the load of the query, depends on which other operators are selected. For example, consider a query q_i with low value and high load. In overload situations, query q_i would surely be rejected in a knapsack auction. But if all of q_i's operators were shared by high value queries, then the effective profit density of q_i (given that we know these high value queries were accepted) could be very high. This dependency between queries makes the mechanism's task much more complex in the case of our continuous query auction than in the case of a knapsack auction. This complexity is illustrated by the fact that there is a polynomial time approximation scheme for the problem of finding the maximum value collection of items to select in a knapsack auction, but even for a special case of our continuous query auction, the densest subgraph problem, it is not known how to approximate the optimal solution to within a polynomial factor in polynomial time [35].

Characterizations of Strategyproofness. A continuous query auction where the only private information is the amount each user values her query is called a single-parameter setting. In single-parameter settings, an allocation mechanism is called *monotone* if every winning bidder remains a winning bidder when increasing her bid. The *critical value* of user i is the value c_i where if the user bids higher than c_i, she wins, but if she bids lower than c_i, she loses. Note that the existence of a critical value for each user is guaranteed by the preceding monotonicity property. It is shown in [66] that a mechanism is bid-strategyproof if and only if it is both *monotone* and each winning user's payment is equal to her *critical value*.

One final auction setting related to our continuous query auction is the *single minded bidders (SMB) auction problem*, studied by Lehmann et al [60]. Each single-minded bidder i is interested in a specific collection S_i from the set of items being auctioned (for a continuous query auction the items being auctioned would be server capacity units and S_i would be the units needed to process the collection of operators in the query q_i). Bidder i has a single, positive valuation v_i for S_i, and has a valuation of zero for any other collection of items. Each bidder specifies not only a bid but also which collection she is interested in, and the set of items is partitioned among the winning bidders. Our CQ auction becomes a SMB auction if both the amount each user values her query and the operators in her query are

private information, and if there is no operator sharing. [16, 60] provide a characterization for strategyproofness in this setting that applies to our setting, even under our more general assumption of operator sharing. Their characterization of a strategyproof mechanism for an SMB auction differs only slightly from the above characterization for any single-parameter setting: the definition of monotonicity is expanded. For a strategyproof SMB mechanism, monotonicity means that not only must a winning bidder remain a winning bidder when increasing her bid, but also must remain a winner when asking for a strict subset of the collection she won. In our CQ admission auction, this corresponds to submitting a query comprised of a strict subset of the operators in the admitted query.

3.1.3 This Work

We present several intuitive greedy by density CQ auction mechanisms. We show that four of these mechanisms, CAF, CAF+, CAT, and CAT+, are bid-strategyproof. Each of these mechanisms has the following form:

- Order queries in decreasing profit density, and then
- admit queries until the server is full.

The intuition is that we wish to accept queries with high valuation to load ratio.

We consider two different definitions of the load of the query. The mechanism CAT assumes that the load of a query is the sum of the loads of its operators. The mechanism CAF assumes that the load of a query is the sum of the fair-share load of its operators, where the *fair-share load* of an operator is the load of the operator divided by the number of queries that share that operator. So intuitively, CAT assumes that there will be minimal or no operator sharing among the accepted queries, while CAF assumes that there will be maximal operator sharing among the accepted queries. The mechanisms CAT and CAF stop once the first query is encountered that will not fit, while the mechanisms CAT+ and CAF+ continue processing the queries hoping to find later, lower load, queries that will fit.

In each of these mechanisms, the price quoted for query q_i is the profit density of a particular rejected query times the load of q_i. For each of CAT and CAF the price for each accepted query is based on the the profit density of the first rejected query. Hence the

Table 1: Summary of the greedy algorithms

Load \ CQ doesn't fit	Stop	Skip
Fair Share Load	CAF	CAF+
Total Load	CAT	CAT+

mechanisms for CAT and CAF are offering a fixed price per unit of server capacity that a query uses. The price for a query q_i in CAF+ and CAT+ is based on the minimum profit density after which q_i would no longer have been accepted (imagine lowering b_i, and hence the profit density, until q_i is not accepted).

In much of the theoretical research on auction problems, the only way that a user can behave strategically is in the setting of her bid. We note that in our CQ admission auction, conceivably a user has other ways in which she could behave strategically. We assume that the estimation of the operator load is done by the server mechanism, and thus the user does not have an opportunity to lie about operator loads. However, a user might conceivably lie about which operators are contained in her query, for example by adding additional operators that are not part of the query she actually desires. A mechanism where users always maximize their payoff by truthfully revealing *all* their private information is called *strategyproof*. In the context of our CQ admission auction then, this means a mechanism is *strategyproof* when both bidding truthfully and submitting only the operators in the query actually desired by the user maximizes the user's payoff. All the bid-strategyproof mechanisms that we consider are also strategyproof.

Finally, we consider a strategic behavior that is well-known in the context of reputation systems like that of eBay and Amazon for rating sellers, buyers and products: a sybil attack. A user who behaves strategically using a sybil attack forges multiple ("fake") identities to manipulate the system. In reputation systems a user might try to boost the reputation of some entity by perhaps adding positive recommendations from false users [45]. In our setting, a sybil attack amounts to creating false identities to submit additional queries that the user

does not need or value in order to manipulate the mechanism. Thus we define a mechanism to be *sybil immune* if a user can never increase her payoff by submitting additional fake, no-value queries. Here we assume that a user's payoff is the aggregate payoff that she gains from the queries of all of her identities. We show that CAT is sybil immune, while the rest of the mechanisms are not. To the best of our knowledge, this is the first time that sybil immunity has been considered in the setting of an auction problem. The notion of sybil immunity can apply to any auction mechanism design problem.

Unfortunately, these greedy by density mechanisms may utterly fail to produce a reasonable profit on some instances. We thus design a bid-strategyproof randomized mechanism called Two Price, that guarantees a profit of at least $OPT - 2h$, where OPT here is the optimal profit that can be obtained by quoting each user the same price (without regard to strategyproofness). Our mechanism is based on the mechanism composition technique from [2]. We compose the greedy mechanism Greedy-by-Value (GV) with the Random-Sampling-Optimal-Price (RSOP) mechanism from [49]. GV simply admits queries in descending bid order until the next query exceeds capacity. The bid-strategyproofness of our mechanism then follows from the fact that GV is a composable mechanism, the fact that RSOP is bid-strategyproof, and the fact that our prices are the maximum of the prices produced by GV and RSOP. Comparing our profit guarantee to the one given in [2] for knapsack auctions: on the positive side, CQ auctions are more general than knapsack auctions, and our guarantee does not degrade with the number of users, and on the negative side, we only compare to the optimal single price profit, which can be much less than the optimal monotone profit [2]. Finally, we show that unfortunately this mechanism is not sybil immune. These game theoretic results are summarized in Table 2.

In the next section we discuss our profit density-based greedy mechanisms and their strategyproofness. In later sections, we discuss a randomized mechanism that yields a provable profit guarantee and evaluate all our mechanisms for sybil immunity. Finally, we present an experimental performance evaluation of our mechanisms, showing that our density-based greedy mechanisms actually outperform the randomized Two Price mechanism in a number of important performance measures.

Table 2: Properties of our proposed auction mechanisms

Mechanism	Sybil Immune	Strategyproof	Profit Guarantee
CAF	×	✓	×
CAF+	×	✓	×
CAT	✓	✓	×
CAT+	×	✓	×
Two Price	×	✓	✓

3.2 GREEDY STRATEGYPROOF MECHANISMS

In order to set the stage, we start by describing a naïve approach that uses the remaining additional load needed to service a CQ for choosing winners and determining payment values. We show how this approach, while it accurately captures the additional load each query will contribute to the total load on the server, is not bid-strategyproof. We then consider the various greedy schemes and discuss their strategyproofness properties.

3.2.1 Agents Chosen by Remaining Load

Consider the following natural mechanism using the above-mentioned greedy scheme for choosing winners. The mechanism first chooses each winner based on a value we define called a query's "remaining load." Then the mechanism charges each winner a payment that also depends on that user's remaining load. We will show that using such a payment scheme is not bid-strategyproof, due to the fact that user's payments are dependent on their bids.

Selecting Winners. We sort the CQs in non-increasing order of priority Pr_i, where $Pr_i = b_i/C_i^R$ and C_i^R is defined as follows.

Definition 3.2.1. *(Remaining Load C_i^R)* The *remaining load* C_i^R of query i is equal to the total load of all the operators of q_i except those operators that are shared with CQs that have already been chosen as winners.

In every iteration through the loop, the algorithm chooses the query with the highest priority and if there is enough remaining capacity in the system to accommodate it, places it in the set of winners. At the end of each iteration, the remaining loads C_i^R as well as the priorities of the yet-unchosen queries are updated. We demonstrate this mechanism with the example in Figure 5(b).

Calculating Payments. We naturally base our first payment mechanism on the known bid-strategyproof k-unit $(k+1)$th-price auction. Recall from Section 3.1.2 that a simple strategyproof mechanism for a k-unit auction is to charge each winning bidder the bid amount of the $(k+1)$th highest bidder. Hence, we define q_{lost} to be the CQ with highest priority that is not a winner. Then, the payment of each winning CQ q_i is calculated as follows: $p_i = C_i^R \cdot b_{lost}/C_{lost}^R$. If the query does not belong to the *winners* list, then the payment is zero.

Remaining Load Algorithm Applied to Example 3.1.1. The initial remaining loads of q_1 , q_2 and q_3 are 5, 6, and 10 respectively. The priorities of q_1, q_2 and q_3 become 11, 12 and 10. During the first iteration of the above algorithm, q_2 is chosen first. Since operator A is chosen as part of q_2, the remaining load of q_1 becomes the load of operator B (just 1 unit) and its priority becomes 55. Consequently, during the second iteration q_1 is chosen. The remaining capacity in the system is 3. During the third iteration, q_3 is chosen however it does not fit in the remaining capacity in the system. As a result, the winners list is composed of q_1 and q_2, and q_{lost} is q_3. As a result, the payments for q_1 and q_2 is \$10 per unit load, which amount to respective payments of \$10 and \$60.

Strategyproofness. The above payment mechanism at first glance seems bid-strategyproof since it is based closely on the well-known bid-strategyproof second-price auction mechanism. However, it is not bid-strategyproof since a winning user i who shares operators with other winning users can gain by bidding lower than her true value. She can strategically bid low enough so that she gets chosen for service *after* the users she shares operators with, but still high enough to win. This will result in a lower remaining load C_i^R and thus in a lower payment.

3.2.2 Agents Chosen by Static Fair Share Load

At this point it has become clear that using remaining load $\left(C_i^R\right)$ for setting payments of users is problematic because of the dependence of these values on the user's bid. Therefore, in this section, we propose using a fixed load that does not change over the course of the winner selection algorithm, and we use that same fixed load to calculate payments.

We define the *static fair share load* as follows.

Definition 3.2.2. Let o_j be an operator that has a load of c_j and is shared among l different CQs, then the static fair share load of o_j per CQ is defined as $c_j^{SF} = c_j/l$. Hence, the *static fair share load* of a CQ q_i is defined as $C_i^{SF} = \sum_{o_j \in Q_i} c_j^{SF}$.

In the following subsections we propose two bid-strategyproof payment mechanisms using the same greedy scheme as in the previous section, but now based on static fair share load: CAF and CAF+.

3.2.2.1 CAF (CQ Admission based on Fair Share) Our first bid-strategyproof mechanism that depends on the static fair share load as defined in Definition 3.2.2 is shown in Algorithm 1.

Selecting winners. Steps 1 through 3 of Algorithm 1 greedily select winners as follows. A priority is assigned to each operator, where the priority is the value-load ratio: $Pr_i = b_i/C_i^{SF}$. Then the list of CQs is sorted in descending order of these priority values. The algorithm admits CQs from the priority list in this order as long as the remaining load C_i^R of hosting the next CQ does not cause system capacity to be exceeded. (Note that the load considered while checking capacity constraints is not the static fair share load.) The algorithm stops as soon as the next CQ does not fit within server capacity.

Calculating payments. Once we have selected the winners, we calculate the payment for each winning user according to steps 4 and 5 of Algorithm 1.

CAF Applied to Example 3.1.1. Since q_1 shares operator A with q_2, C_1^{SF} is 3 and C_2^{SF} is 4. During the first iteration of CAF, the priorities of q_1, q_2 and q_3 are 18.34, 18, and 10. As a result, CAF chooses q_1 first and then q_2. Again, q_3 is q_{lost}. Thus the payments for q_1 and q_2 are \$10 per unit load, which amount to respective payments of \$30 and \$40.

Algorithm 1 Our basic fair share mechanism (CAF). **Input:** A set of queries with their static fair share loads C_i^{SF} and their corresponding bids b_i. **Output:** The set of queries to be serviced and their corresponding payments.

1. Set priority Pr_i to b_i/C_i^{SF} for each query i.
2. Sort and renumber queries in non-increasing Pr_i so that $Pr_1 \geq Pr_2 \geq \ldots \geq Pr_n$.
3. Add the maximal prefix of the queries in this ordered list that fits within server capacity to the winner list.
4. Let *lost* be the index of the first losing user in the above priority list.
5. Charge each winner i a payment of $p_i = C_i^{SF}(b_{lost}/C_{lost}^{FS})$. Charge all other users 0.

Strategyproofness. We prove the following theorem by using the monotonicity and critical value characterization of bid-strategyproof mechanisms for any single-parameter setting (see Section 3.1.2).

Theorem 3.2.3. *The CAF mechanism is bid-strategyproof.*

Proof. The CAF winner selection is clearly monotone: any winning bidder could not become a loser by increasing her bid since she will only move up in the priority list by doing so. The CAF payments are also equal to the users' critical values. If user i bids $b_i' < C_i^{SF}(b_{lost}/C_{lost}^{SF})$, then we would have $b_i'/C_i^{SF} < b_{lost}/C_{lost}^{SF}$ and we know that both user i and user *lost* could not fit together on the server with the other winners, so user i will become a loser. □

We also note that CAF is not just bid-strategyproof, but strategyproof. This results from the fact that the characterization for SMB auctions in [60] carries over to our setting (see Section 3.1.2), and that CAF satisfies their additional monotonicity requirement that when a winning bidder asks for only a subset of the operators in her query, she still wins.

3.2.2.2 CAF+: An Extension to CAF Selecting winners. CAF+ extends CAF by allowing the algorithm to continue until there are no unserviced CQs left that will fit in the remaining server capacity. While CAF stops as soon as it encounters a query whose load exceeds remaining capacity, CAF+ skips over any queries that are too costly, continuing

onto more light-weight queries in the priority list. (See Algorithm 2.)

Calculating payments. The algorithm calculates the payment of each winning user (or serviced query) based on each user's *movement window*. Intuitively, the movement window of a winning user is the amount of freedom the user has to bid lower than her actual valuation without losing. A more formal definition follows.

Definition 3.2.4. In CAF+ every query that is selected to be serviced has a *movement window*. A user's movement window is defined as a sublist of the complete list of queries ordered in descending priority $Pr_i = b_i/C_i^{SF}$. We will refer to this complete list as the *priority list*. The movement window of winning user i begins with the user just after user i in the priority list, and ends at the first user j in the priority list that both comes after i and satisfies the following property: if user i's bid was changed so that it directly followed the position of user j in the priority list, CAF+ would no longer choose query i as a winner. If such a user j does not exist, then user i's movement window spans the entire remainder of the priority list.

Definition 3.2.5. For each winning query q_i, $last(i)$ is defined to be the first query which is outside q_i's movement window. If there are no queries remaining outside the movement window of q_i, then $last(i)$ is set to *null*.

The payment in CAF+ (Algorithm 2) is calculated for each query after the set of queries to be serviced is determined. To specify the payment of each winner i, the algorithm first calculates the identity of $last(i)$. Then the payment for the selected query is defined as $p_i = C_i^{SF} \cdot b_{last(i)}/C_{last(i)}^{SF}$. If user i's movement window included all remaining queries in the priority list, i.e., if $last(i) = null$, then the payment of user i is 0.

Strategyproofness. The proof that CAF+ is bid-strategyproof is similar to that of Theorem 3.2.3; we again use the characterization of bid-strategyproofness given in [66].

Theorem 3.2.6. *The CAF+ mechanism is bid-strategyproof.*

Proof. The CAF+ winner selection is monotone: any winning bidder could not become a loser by increasing her bid since she will only move earlier in the priority list by doing so. The CAF+ payments, by definition of each user's movement window (Definition 3.2.4) are precisely equal to the minimum value the user must bid in order to remain a winner. □

Algorithm 2 Our aggressive fairshare mechanism (CAF+). **Input:** A set of queries with their static fair share loads C_i^{SF} and their corresponding bids b_i. **Output:** The set of queries to be serviced and their corresponding payments.

1. Set priority Pr_i to b_i/C_i^{SF} for each query i.
2. Sort and renumber queries in non-increasing Pr_i so that $Pr_1 \geq Pr_2 \geq \ldots \geq Pr_n$.
3. For $i = 1 \ldots n$, add user i to the winner list if doing so does not exceed capacity.
4. For each winner i, calculate $last(i)$, as defined in Definition 3.2.5.
5. Charge each winner i a payment of $p_i = C_i^{SF}(b_{last(i)}/C_{last(i)}^{FS})$. Charge all other users 0.

As with CAF, we note that CAF+ is not only bid-strategyproof, but strategyproof. The reasoning is the same as for CAF (see Section 3.2.2.1).

3.2.3 Agents Chosen by Total Load

Because the "fairshare" based mechanisms described above are vulnerable to certain types of user manipulation (see Section 3.4), we propose two more robust mechanisms. These mechanism are exactly analogous to the mechanism from Section 3.2.2, except that we replace every incidence of the static fairshare load C_i^{SF} with that total load $C_i^T = \sum_{o_j \in Q_i} c_j$. Thus we have two mechanisms.

- **CAT** (CQ Admission based on Total load): analogous to CAF described in Section 3.2.2.1.
- **CAT+**: analogous to CAF+ described in Section 3.2.2.2.

CAT Applied to Example 3.1.1. In example 3.1.1 C_1^T, C_2^T and C_3^T are 5, 6 and 10 units. Thus Pr_1, Pr_2 and Pr_3 are 11, 12, and 10. Consequently, CAT chooses q_1 and q_2 to be serviced. The payments for q_1 and q_2 are \$10 per unit load, which amount to respective payments of \$50 and \$60.

It is easy to verify that the proofs of bid-strategyproofness carry over to these modified versions of the algorithms and payments. We therefore have the following two theorems.

Theorem 3.2.7. *The CAT mechanism is bid-strategyproof.*

Theorem 3.2.8. *The CAT+ mechanism is bid-strategyproof.*

As with CAF and CAF+, we note that both CAT and CAT+ are not only bid-strategyproof, but strategyproof.

3.3 A PROFIT GUARANTEE

In this section we present a bid-strategyproof randomized mechanism that is competitive (in expectation) with the best optimal constant pricing mechanism with respect to the goal of maximizing profit. A *constant pricing* mechanism as defined in [2], is any mechanism (strategyproof or not) where the users are all charged the same price, call it p, and those who bid strictly higher than p are winners, those who bid strictly lower than p are losers, and those who bid equal to p may be designated winners or losers arbitrarily by the mechanism. Winners must all pay p and losers pay 0. *Profit* is defined to be the sum of the payments that the mechanism charges or receives from the users.

A *constant pricing* mechanism is *valid* if all winners fit within server capacity, and so we will only consider valid constant prices. *Optimal constant pricing profit* then refers to the maximum possible profit that can be attained from any valid constant pricing mechanism (strategyproof or not). We choose to focus on constant pricing optimality in this work because with the shared processing of queries in our problem, other standard profit benchmarks seem difficult to compete with. Two other natural profit benchmarks include optimal pricing per unit load and optimal monotone pricing, both of which generalize optimal constant pricing and were discussed in the context of Knapsack Auctions in [2]. But because of our shared processing between queries, the processing load required of each query is not clear cut. Hence both proportional and monotone pricing definitions become fuzzy.

3.3.1 A Randomized Mechanism

In this section we show that by only using two distinct prices, under the assumption that the users all have distinct valuations, we are able to find a bid-strategyproof mechanism

that approximates optimal constant pricing profit. We show however that there is a trade-off between the run-time of the mechanism and its profit. We first present a mechanism that runs in time exponential in the number of duplicate valuations, then explain how a polynomial time version of it gives a weaker profit guarantee.

We refer to our mechanism as the *Two-price Mechanism* (see Algorithm 3). The first phase of the mechanism (Steps 1 and 2) follows our greedy scheme (using user valuations), the second phase (Step 3) is an exhaustive search that gives the potential exponential running time in terms of number of duplicate valuations, and the last phase (Steps 4 through 6) contains the randomization and is essentially identical to the Random Sampling Optimal Price auction of [49, 51].

Note that in Step 3 of the mechanism we run an exhaustive search on all possible subsets of the critical set of queries with duplicate valuations. The possibility of sharing of server capacity between queries is what requires us to take this potentially arduous step, as the problem of optimally determining which subset of queries to admit in the face of such sharing seems hard to approximate.

3.3.1.1 Strategyproofness A randomized mechanism is *bid-strategyproof in expectation* if for every user i, the expected payoff for user i is maximized when user i bids her true valuation v_i [66]. A randomized mechanism is *bid-strategyproof in the universal sense* if it is not only bid-strategyproof in expectation, but it is also "ex post" bid-strategyproof. That is, regardless of the outcome of the randomness, users always maximize their payoff by bidding their true valuations. In other words, a universally bid-strategyproof randomized mechanism is a probability distribution over bid-strategyproof deterministic mechanisms [66].

To prove that our mechanism is bid-strategyproof in the universal sense, we depend on some existing results. In [2], Aggarwal and Hartline define *mechanism composition* as follows.

Definition 3.3.1. Given two mechanisms M_1 and M_2, define the composite mechanism $M_1 \circ M_2$, as:

1. Simulate M_1 and let H be the set of winners.

Algorithm 3 *Two-price mechanism.* **Input:** Set of n queries and corresponding user valuations $v_1 \ldots v_n$. **Output:** Set of winners and their corresponding payments.

1. Sort and renumber the queries in order of decreasing valuation, so $v_1 \geq v_2 \geq v_3 \geq \ldots \geq v_n$, breaking ties arbitrarily.
2. Let H be the ordered set of queries that comprise the maximal prefix of queries from this sorted list that fits within our server capacity. Let L be the ordered set of losers (remaining queries not chosen for H) and let v_L be the valuation corresponding to the first query in L.
3. If the last query in H has valuation v_L, the set of queries in H must be adjusted as follows. Let D be the set of all users with valuation equal to v_L, and let d be the cardinality of D. Let $H' = H - D$. Let D^* be the largest subset of D that fits within capacity along with H'. Redefine $H = H' + D^*$
4. Partition the users from H evenly into two sets, A and B, uniformly at random. Renumber queries separately in each set as in step 1. I.e., $v_1 \geq v_2 \geq \ldots \geq v_a$ for the a queries in set A, and $v_1 \geq v_2 \geq \ldots \geq v_b$ for the b queries in set B, again breaking ties arbitrarily.
5. Calculate the optimal constant price profit of each set of queries: $OPT(A) = \max_{i \in A} iv_i$ and $OPT(B) = \max_{i \in B} iv_i$. Let $k = \arg\max_{i \in A} iv_i$ and let $p_A = v_k$. Similarly, let $j = \arg\max_{i \in B} iv_i$ and let $p_B = v_j$.
6. Use the price p_A to determine the winners from set B and use the price p_B to determine the winners from set A. Specifically, the winners from set B are those users whose valuations are greater than p_A, and these winners are each charged a payment of p_A. Similarly determine winners and payments for users in set A.

2. Simulate M_2 on the set H.
3. Offer a price to each winner of Step 2 that is the maximum of the price she is offered by M_1 and M_2.

They then define a mechanism to be *composable* if it is both bid-strategyproof and the set of chosen winners does not change as any winning user varies her bid above her critical value.

Finally, they prove the following lemma.

Lemma 3.3.2. *([2]) If mechanism M_1 is composable and mechanism M_2 is bid-strategyproof then the composite mechanism $M_1 \circ M_2$ is bid-strategyproof.*

To prove the following theorem, we use Lemma 3.3.2.

Theorem 3.3.3. *The* Two-price *mechanism is bid-strategyproof in the universal sense.*

Proof. We use Lemma 3.3.2. Let M_1 be the mechanism defined by steps 1 and 2 of the *Two-price* mechanism along with its corresponding critical payments: each user pays an amount equal to v_L, the highest losing bid as defined in Step 2 of the algorithm. This is bid-strategyproof as it is equivalent to a standard k-Vickrey auction (see Section 3.1.2), and it is composable since any winner varying her price above the highest losing bid does not change the set of winners. We let M_2 be the mechanism defined by the remaining steps of the Two-price mechanism. Note that M_2 is a *randomized bid-independent auction* (as defined in [37]). Theorem 2.1 in [37] states that an auction is universally bid-strategyproof if and only if it is bid-independent. Because the payments set by M_2 will always be higher than those set by M_1, we can invoke Lemma 3.3.2 to conclude our entire mechanism is bid-strategyproof. □

Note that because the Two-price mechanism allocates winners and sets payments entirely independent of each query's load, it is not only bid-strategyproof, but strategyproof.

3.3.1.2 Competitiveness We assume user valuations range from 1 to h and we use OPT to refer to the optimal constant pricing profit.

Theorem 3.3.4. *The expected profit of the* Two-price *mechanism is at least $OPT - 2h$.*

Proof. Let $n(S, p)$ refer to the number of users in set S whose valuations are p or higher. Then the Two-price mechanism's expected profit can be expressed as

$$TP = \mathbf{E}[n(A, p_B)p_B + n(B, p_A)p_A].$$

Observe that

$$\begin{aligned} \mathbf{E}[n(A, p_B)p_B] &\geq \mathbf{E}[n(B, p_B)p_B] - p_B & (3.1) \\ &\geq \mathbf{E}[n(B, p^*)p^*] - p_B, & (3.2) \end{aligned}$$

where p^* is the optimal constant price if our input was the set H and the second inequality holds by definition of p_B. Then observe that

$$\mathbf{E}[n(B, p^*)p^*] = \frac{n(H, p^*)p^*}{2} = \frac{OPT(H)}{2} = \frac{OPT}{2} \tag{3.3}$$

where OPT(H) refers to the optimal solution if the input is only the queries in H, and the last equality holds because any *valid* optimal constant price can be no less than the minimum valuation of any user in H. Putting these together and upper bounding p_B with h gives us

$$\mathbf{E}[n(A, p_B)p_B] \geq \frac{OPT - 2h}{2}.$$

By symmetric arguments for $\mathbf{E}[n(B, p_A)p_A]$ and linearity of expectation we can then conclude $TP \geq OPT - 2h$. □

We can also show that eliminating Step 3 of the *Two-price* mechanism, yielding a polynomial-time algorithm, gives a weaker profit guarantee parameterized by the maximum number of duplicate valuations. Specifically, if there are d valuations in the input that are identical, then the profit guarantee decreases to $OPT - dh$. The analysis is similar to the proof of Theorem 3.3.4, the difference being that we can no longer claim $OPT(H) = OPT$ as in equation (3.3). Eliminating Step 3 of the mechanism means we can only say that H has at least 1 of the queries in the set D, while OPT has at most $d - 1$ of the queries in D, so we instead obtain $OPT(H) \geq OPT - (d - 2)h$. Combining this with the rest of the analysis, which remains the same, gives us the following result.

Theorem 3.3.5. *Expected profit of the polynomial-time mechanism defined by the* Two-price *mechanism without Step 3 is at least* $OPT - dh$.

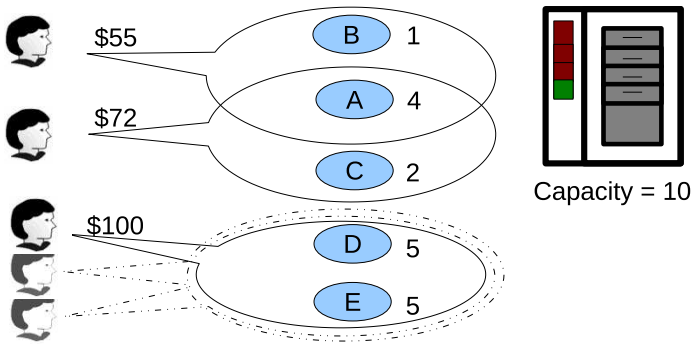


Figure 6: This figure illustrates the third user from Example 3.1.1 perpetrating a sybil attack by forging two additional fake users. The real queries are indicated in solid lines while the fake queries are indicated in dashed lines. The presence of these fake queries creates the illusion that user 3's operators are in higher demand, which could conceivably influence the mechanism to either charge the third user less, or service her when she would not have otherwise been serviced.

3.4 SYBIL ATTACK

In this section we consider a strategic behavior that is well-known in the context of reputation systems like that of eBay and Amazon for rating sellers, buyers and products: a sybil attack. A user who behaves strategically using a sybil attack forges multiple ("fake") identities to manipulate the system. In reputation systems a user might try to boost the reputation of some entity by perhaps adding positive recommendations from false users [45]. In our setting, a sybil attack amounts to creating false identities to submit additional queries that the user does not need or value in order to manipulate the mechanism. (See Figure 6.) Thus we define a mechanism to be *sybil immune* if a user can never increase her payoff by submitting additional fake, no-value queries. We make the natural assumption that if a fake query is chosen to be serviced, the sybil attacker is responsible for making the fake query's payments, so a user's payoff is the aggregate payoff that she gains from the queries of all of her identities.

We will show here that CAT is sybil immune, while the rest of the mechanisms are not. To the best of our knowledge, this is the first time that sybil immunity has been proposed, and we note that the notion of sybil immunity can apply to any mechanism design problem.

Definition 3.4.1. We define a mechanism to be *vulnerable* to sybil attack if there exists an input instance where there is a user who can increase her payoff by perpetrating a sybil attack.

Definition 3.4.2. We define a mechanism to be *universally vulnerable* to sybil attack if in every input instance, every user has a way to improve her payoff by perpetrating a sybil attack.

In this section we show that while some of our mechanisms are extremely vulnerable to sybil attack, our CAT mechanism is robust to sybil attack.

3.4.1 Attacks Against the Fair Share Mechanisms

Unfortunately, our proposed fair-share schemes of Section 3.2.2 are vulnerable to sybil attack. A user i can employ the following strategy using a sybil attack to improve her payoff: simply create fake users with negligible valuations whose queries share operators with q_i. A sybil attack of this kind will lower the attacker's fair share load, improving her ranking and enabling her to be selected as a winner while simultaneously decreasing her payment to an affordable amount. Note that it is always possible for the attacker to set her fake users' valuations low enough so that they are not in danger of being selected as winners, and hence will require no additional payment from the attacker.

Indeed, we can prove that in *any* given instance of the CQ admission problem, *any* user can gain from employing a sybil attack against our fair share mechanism.

Theorem 3.4.3. *Both the CAF or CAF+ mechanisms are* universally vulnerable *to sybil attack.*

Proof. Consider any given user i. We have two cases: either user i is a winner under the mechanism in question (either CAF or CAF+), or i is a loser.

Case 1. If i is a loser, then i can gain by perpetrating a sybil attack as follows. Let j be a winning user such that $j = \arg\max_k Pr_k$. Choose the number s of forged queries to be such that $v_i/(C_i^T/s) > v_j/C_j^{SF}$. User i creates s fake users, each with query identical to q_i. Doing so makes $C_i^{SF} \leq C_i^T/s$, which by choice of s means $Pr_i = v_i/C_i^{SF} > v_j/C_j^{SF} = Pr_j$. Since j was the winner with highest priority, and all users have total load at most 1 by assumption, i is now a winner. We can see that i's payoff has improved because when i was a loser her payoff was 0, while now it is $v_i - p_i = v_i - C_i^{SF} v_j/C_j^{SF} > 0$. User i can also ensure that she makes no payments for the fake queries she created by setting their valuations to be sufficiently small so that they all lose.

Case 2. If i is a winner, then i can gain by using a sybil attack to reduce i's payment, p_i. If i simply creates one fake user whose query is identical to q_i, C_i^{SF} will be reduced, which reduces $p_i = C_i^{SF}(v_{lost}/C_{lost}^{SF})$. Again, user i chooses the fake user's valuation to be a sufficiently small value to ensure that the fake query's priority is low enough to not get serviced. □

3.4.2 Attacks Against the Total Load Mechanisms

In contrast to this vulnerability of our fair share mechanisms, the total load payment mechanisms (CAT and CAT+), described in Section 3.2.3, seem at first to be robust to such attacks. While we've seen that a user's fair share can easily be reduced by creating fake identities, a user's total load is not dependent on the number of other users sharing her load, and therefore CAT and CAT+ should not (at least at first glance) be prone to such sybil attack strategies.

Definition 3.4.4. We say that a mechanism is *immune to sybil attack* if for every input instance, no user can increase her payoff by perpetrating a sybil attack (i.e., it is not vulnerable). We also use the term *sybil immunity* to refer to this property.

However, one of our total load mechanisms is not immune to sybil attack. We begin by giving the following characterization of sybil immunity. A mechanism is sybil immune if and only if both of the following properties hold:

1. The arrival of additional queries will never cause a loser to become a winner with positive

payoff.

2. If the arrival of additional queries reduces a winner's payment by δ, the additional queries that become winners must be charged a total of at least δ by the mechanism.

We now show that CAT+ is vulnerable to sybil attack because it does not satisfy property 1 of the above characterization.

Theorem 3.4.5. *For the CQ admission problem, CAT+ is vulnerable to sybil attack.*

Table 3: An example of a sybil attack that beats CAT+. User 2 is a sybil attacker, creating a fake query that appears to the system as "user 3". Here, ϵ represents an arbitrarily small positive value.

User	1	2	"3"
v_i	100	89	$100\epsilon + \epsilon$
C_i^T	1	0.9	ϵ
Pr_i	100	< 100	> 100
Round 1	100	< 100	picked
Round 2	exceeds cap.	picked	picked
Payments p_i	0	0	$\leftarrow 100\epsilon$
Payoffs	0	$89 - 100\epsilon$	N/A

Proof. Consider the example in Table 3, in which a sybil attacker defeats our total load algorithm, CAT+. In this example, if user 2 does not perpetrate the attack, user 1 will get chosen for service, and then server capacity will be reached, so user 2 would not get serviced. Whereas when user 2 introduces the fake "user 3," she is able to trick the system into choosing her instead of user 1. □

Note that in this kind of sybil attack, the danger for user 2 (the attacker) is that when the fake "user 3" was chosen for service, user 2 had to make user 3's payment. Hence user 3's fake valuation and fake load had to be carefully chosen by user 2 so that user 2 found paying user 3's fee worthwhile. (Recall from Section 3.2.3 that payment of a winning user i

is calculated as $C_i^T v_{lost}/C_{lost}^T$, so in our example, that makes $p_3 = 100\epsilon$). In this particular instance, user 2 had no payment of her own to pay because there are no users that have lower priority than user 2. This makes paying "user 3"'s payment affordable to user 2.

The good news is: our total load mechanisms are not always bad. First, while our fair share mechanisms are *universally vulnerable* to attack, there are instances under the total share mechanism that are robust to sybil attack. Second, and more notably, the CAT mechanism is immune to sybil attack. In fact we can make an even stronger claim. Thus far in our discussion of sybil attacks, we have been considering sybil attack in isolation from bid-strategyproofness. However, it is possible that a user can use a sybil attack *in conjunction* with lying about her valuation in order to increase her payoff. This possibility raises the question of whether adding sybil attacks to each user's set of possible strategies has removed our mechanism's bid-strategyproofness.

It turns out that our CAT mechanism remains bid-strategyproof even if we allow sybil attacks, *and* it remains immune to sybil attack, even if we allow users to lie about their valuations.

Definition 3.4.6. We define a mechanism to be *sybil-strategyproof* if no user can improve her payoff by either lying about her valuation, perpetrating a sybil attack, or doing both simultaneously.

We now give a characterization of sybil-strategyproof mechanisms. A mechanism is sybil-strategyproof if and only if both of the following properties hold:

1. It is bid-strategyproof.
2. The arrival of additional users (e.g., via a sybil attack) cannot decrease anyone's critical value by an amount more than the total payment charged to the additional users.

We use the above characterization to prove that CAT is sybil-strategyproof.

Theorem 3.4.7. *For the CQ admission problem, the CAT mechanism is sybil-strategyproof.*

Proof. Property 1 of the characterization of sybil-strategyproofness is satisfied since we have already seen from Theorem 3.2.7 that CAT is bid-strategyproof. To satisfy property 2 of

the characterization, it is sufficient to show that adding additional users to the instance does not decrease any user's critical value. Consider any user i. In the CAT mechanism, user i has critical value $p_i = C_i^T(v_{lost}/C_{lost}^T)$. (Recall that *lost* is defined to be the user with highest priority not selected to be serviced by CAT.) Since the arrival of additional users cannot change C_i^T, we need only show that the arrival of additional users cannot decrease $v_{lost}/C_{lost}^T = Pr_{lost}$.

We proceed by induction on the number of additional users that arrive. Assume inductively that introducing the first k users cannot reduce Pr_{lost}. Define $lost(k)$ to be *lost* in the instance that includes only the first k fake users, and let $lost(k+1)$ be *lost* in the instance that includes the first $k+1$ users. We must show that $Pr_{lost(k+1)} = v_{lost(k+1)}/C_{lost(k+1)}^T \geq Pr_{lost(k)} = v_{lost(k)}/C_{lost(k)}^T$.

Consider user j, the $(k+1)$th newly arriving user. If user j has priority $Pr_j \leq Pr_{lost(k)}$, then $lost(k+1) = lost(k)$ and hence $Pr_{lost(k+1)} = Pr_{lost(k)}$. If $Pr_j > Pr_{lost(k)}$ then $Pr_{lost(k+1)} \geq Pr_{lost(k)}$.

Hence the critical value of user i cannot be decreased by the arrival of additional users.

□

3.4.3 Attacks Against the Randomized Mechanism

Our randomized *Two-price* mechanism, however, is not immune to sybil attack. We prove this fact by showing that the mechanism violates property 2 of our characterization of sybil immunity: a winning user can reduce her payment (in expectation) by introducing fake queries such that the fake queries incur less expected total charges than the amount her payment was reduced by.

Theorem 3.4.8. *The* Two-price *mechanism is vulnerable to sybil attack.*

Proof. Consider the following input instance. User 1 has valuation $b > 1$ and users 2 and 3 have valuation $c > 1$, where $b > c$ and c is an integer. Suppose all three users make it past the first three Steps of the mechanism into the set H with some positive capacity to spare. (Assume all other users have very large size and valuation less than 1 and were thus placed in set L.)

The mechanism will always charge user 1 a price of c, regardless of the randomness, giving user 1 a payoff of $b - c$. However, user 1 can benefit from sybil attack as follows. User 1 creates $2c - 3$ fake queries, each with valuation $1 + \epsilon$, and each with size small enough that they are also placed in the set H. We consider three cases.

Case 1: users 2 and 3 are in the opposite partition from user 1. Without loss of generality, assume user 1 is in set A and users 2 and 3 are in set B. In this case, it is impossible for the fake users to change the payoff of user 1: $c - 2$ of them are placed in the set B, but $(1 + \epsilon)c < 2c$, so p_B remains at c as before. And none of the fake users are winners.

Case 2: users 2 and 3 are in opposite partitions from one another. Without loss of generality, assume users 1 and 2 are placed in set A and user 3 is placed in set B. Now, since $(1+\epsilon)c > c$, the price p_B becomes $1+\epsilon$, giving user 1, our sybil attacker, a lower price. The fake users in set A also must pay $1 + \epsilon$, but there are only $c - 2$ of them, so user 1's net payoff becomes $b - (1 + \epsilon) - (c - 2)(1 + \epsilon) > b - c$, for small enough ϵ.

Case 3: users 1, 2 and 3 are in the same partition. Assume they are all placed in set A. Then user 1's new price is again $1 + \epsilon$. Again, even after making the payments for her fake users, users 1's payoff has improved.

Thus, overall, user 1's expected payoff improves due to her sybil attack. □

Finally, we note that even if we modify Step 4 of the mechanism so that each query is placed in set A or B based on independent coin flips (so that H may not be evenly partitioned), the mechanism is still vulnerable to sybil attack. Again, the vulnerability is due to a violation of property 2 of our characterization of sybil immunity. Consider the instance where user 1 has valuation b and n_c users (which get placed into H along with user 1) all have a valuation of $c < b$. Set server capacity to be equal to the size of the users in H.

User 1 creates a fake user with valuation $d = c + \epsilon$, with size equal to the combined size of all the users with valuation c, kicking them out of H. While before user 1 was charged c with probability $1 - (1/2)^{n_c}$ and 0 with probability $(1/2)^{n_c}$, now that only user 1 and the fake user are in H, user 1 and her fake user is charged 0 with probability $1/2$, and d with probability $1/2$. For choice of ϵ that ensures $d/2 < c(1 - (1/2)^{n_c})$, user 1's expected payoff has decreased.

3.5 EXPERIMENTAL EVALUATION

In this section, we experimentally demonstrate the behavior of the different proposed auction-based admission control mechanisms. First we present the experimental setup. Then we discuss the results.

3.5.1 Experimental Platform

Mechanisms. We implemented all the proposed admission control mechanisms in Java, including the strategyproof GV (Greedy by Valuation) mechanism (described in Section 3.2). We also implemented the optimal constant pricing profit (OPT_C) algorithm, described in Section 3.3.

Metrics. For each mechanism, we measured the following performance metrics:

- Profit: the sum of the payments of the admitted queries.
- Admission rate: The percentage of queries admitted.
- Total user payoff: the sum of the valuations (bids) of the admitted queries minus the payments. Total user payoff can be seen as an indication of total user satisfaction under each mechanism.
- System utilization: the used capacity of the server.
- The runtime for each mechanism.

The reported results are the average of running each algorithm on 50 different sets of workload. Note that, for clarity, our figures do not show GV or OPT_C as they echo the behavior of Two-price in all experiments.

Workload. We summarize the workload parameters in Table 4. We generated 50 sets of workload for four different system capacities. Each set contains a number of different input instances. An input instance consists of users' queries along with their bids, and is parameterized by:

- A system capacity.
- Maximum degree of sharing: The degree of sharing of an operator is the number of queries that share a single operator, and the maximum is taken over all the operators.

Table 4: Workload Characteristics

Number of workload sets	50
Number of queries	2000
Number of operators	$700 \sim 8800$
Max Degree of Sharing	$[1 - 60]$ - Zipf, skewness: 1
Maximum Bid	100 - Zipf, skewness: 0.5
Maximum Operator Load	10 - Zipf, skewness: 1
System Capacity	5K-10K-15K-20K

We varied the maximum degree of sharing from 1 to 60. We keep the average query load the same throughout a workload set, while varying the maximum degree of sharing. To achieve this, we generate a workload with the highest maximum degree of sharing (i.e. 60) and then gradually split the operators of the highest degree and distribute the resulting operators into other varying degrees within a workload. For example, to generate an input instance of maximum degree of sharing 7, using the input instance of max degree of sharing 8, if there were 100 operators with degree 8, we split each one of them to degrees 4,2,1,1 (four operators). This will generate 400 new operators with the same load as the original operators. The queries associated with that operator will be distributed among the resulting operators. Each input instance consists of 2000 queries and between 700 and 8800 operators (the number of operators decreases as the degree of sharing increases).

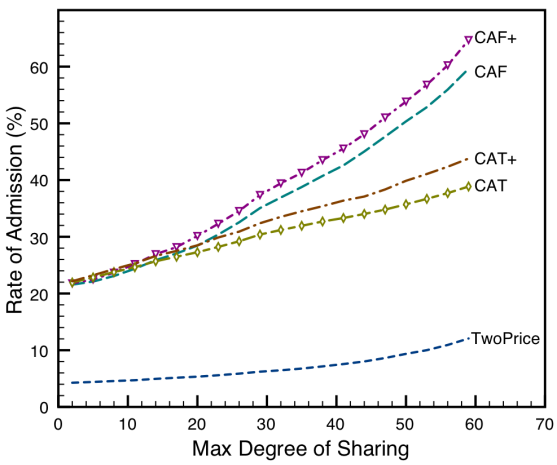
The bids of each query are randomly generated according to a Zipf distribution with maximum bid value set to 100 and skewness parameter set to 0.5. The load of each operator is also randomly generated according to a Zipf distribution with the maximum operator load set to 10 units and skewness parameter set to 1. Operators are assigned to queries randomly, where for each operator, the number of queries sharing it is drawn from a Zipf distribution with skewness parameter set to 1 and the maximum degree of sharing changes from 1 to 60.

3.5.2 Experimental Results.

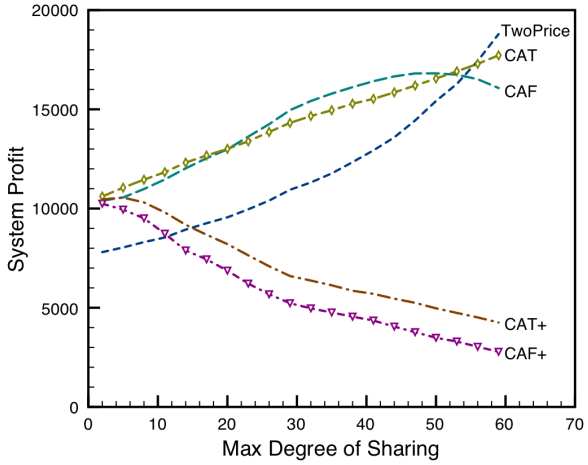
Figure 7(a) shows the percentage of admitted queries as the degree of sharing ranges from 1 to 60, for a system with capacity 15,000. All mechanisms admit more queries as the degree of sharing increases. This is due to the fact that the mechanisms are able to take advantage of the shared processing between queries, so more queries can be serviced using the same system capacity. Two-price always admits a smaller percentage of the queries than the density-based mechanisms (CAF, CAF+, CAT, CAT+) because it chooses queries by user bid only, without regard to query load.

Interestingly, profit (in Figure 7(b)) does not follow the same trend. CAF and CAT are the best for profit, as they do not admit queries as greedily as CAF+ and CAT+ do, which means the prices they charge admitted queries are much higher than CAF+ and CAT+. The profit of CAF+ and CAT+ decrease as degree of sharing increases because they are simply admitting so many queries (as sharing increases) that the prices they are charging admitted queries continues to be driven downward. Due to the fact that queries are selected in decreasing order of density and charged a per-unit price equal to the per-unit bid of the first losing query, very few queries means higher prices, more queries means lower prices. The Two-price mechanism provides profit that consistently improves as degree of sharing increases because its profit is close to the optimal constant pricing profit, which only improves as the number of queries that can fit within capacity increases. At the point where Two-price crosses over CAF and CAT, we observe the same phenomenon that caused decreasing profit in CAF+ and CAT+. At the crossover point, CAF and CAT begin to admit such a high rate of queries that the prices they are charging are being driven dramatically downward (remember, query valuations are drawn from a skewed distribution), reducing overall profit faster than the gain in profit from admitting more queries. The profit of CAF in particular begins to really dive, as the payments are an increasing function of each query's fair share load, which also shrinks as the degree of sharing increases.

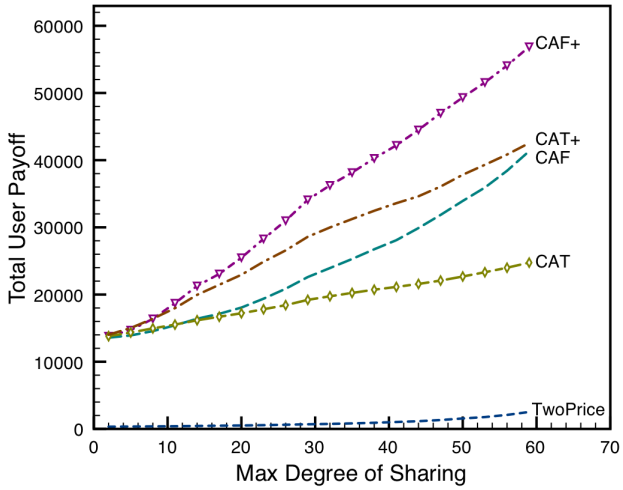
With respect to maximizing total user payoff (Figure 7(c)), the density based mechanisms always perform better than Two-price because they are able to admit more queries and satisfy more customers. CAF+, of course, has the highest payoff because not only are the most



(a) System Capacity = 15,000



(b) System Capacity = 15,000



(c) System Capacity = 15,000

Figure 7: Figure 7(a) shows the percentage of queries serviced under each mechanism. Figure 7(b) shows the profit earned by each mechanism. Profit is the sum of the payments of the admitted queries as they were calculated by the respective mechanisms. Figure 7(c) shows total user payoff, which can be interpreted as a measure of total user-satisfaction. A user's payoff is defined as her valuation minus her payment. Seen here is the sum of winning users' payoffs.

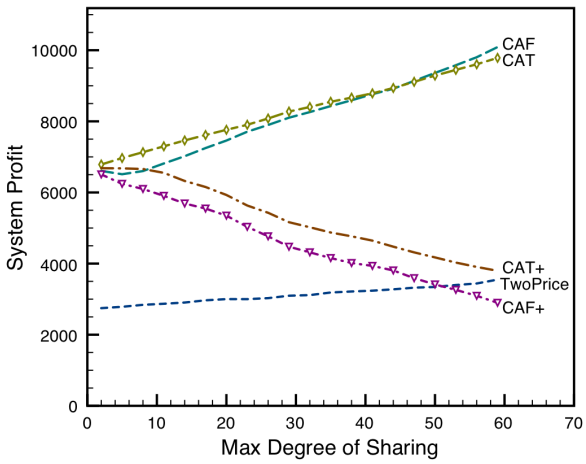
queries admitted under CAF+, but users are only paying for their fair share load, rather than for their total load. As the degree of sharing increases, CAF begins to overtake CAT+ in total user payoff because fair share load per user is decreasing, which decreases payments, increasing payoffs. Each query's total load on the other hand, remains constant as the degree of sharing increases.

In terms of utilization, we found that all proposed mechanisms admit queries so as to utilize more than 98 percent of the system capacity, except for Two-price which utilizes between 96 percent and 98 percent.

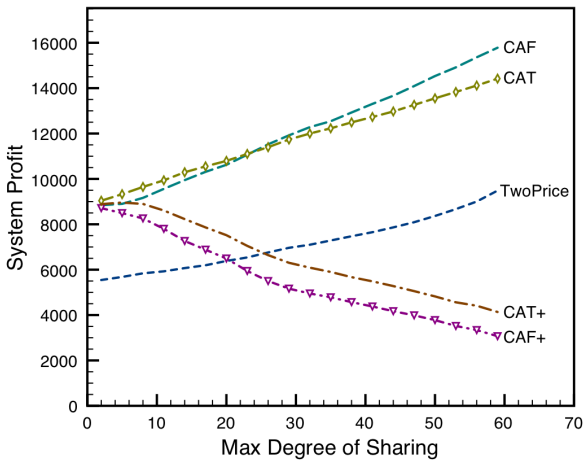
In Figure 8, we show the system profit for three other system capacities. As system capacity increases, it is apparent that the crossover points (between CAF+, CAT+ and Two-price and between CAF, CAT and Two-price) are shifted to the left, to lower degrees of sharing. Indeed, as capacity increases, the picture as a whole seems to shift and scale downward to the lower end of max degree of sharing. When system capacity is close to the total query demand and sharing is high, the Two-price mechanism has clearly overtaken all the density mechanisms for highest profit. As described above, this is due to the fact that so many of the queries are being serviced by the density mechanisms, driving down the prices being charged.

We list the average runtime performance of each mechanism over all workloads with 2000 queries and capacity 15K in Table 5. As a baseline, we also implemented a randomly admitting algorithm, which picks queries at random and stops at the first query that does not fit in the remaining capacity. The algorithms ran on an Intel Xeon 8 core 2.3GHz, with 16GB of RAM. The mechanisms only utilized one core. It is clear that the more aggressive mechanisms (CAF+ and CAT+) cannot scale compared to the simple ones. We note here that even though the density based mechanisms' runtime is only three to seven times more than the baseline random algorithm, they provide strategyproofness, and moreover CAT also provides sybil-immunity.

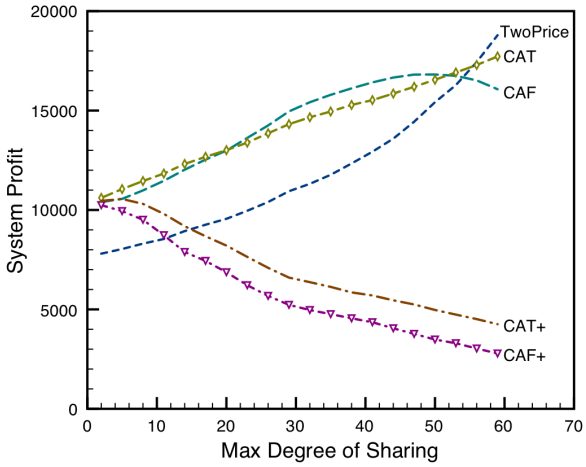
Manipulation of the System. Finally, we evaluate CAR for profit both in a setting where users are being truthful about their valuations, and in a setting where they strategize and bid less than their true valuations (i.e., "lie"). Since CAR is the only mechanism that is not strategyproof, such lying under CAR is to be expected.



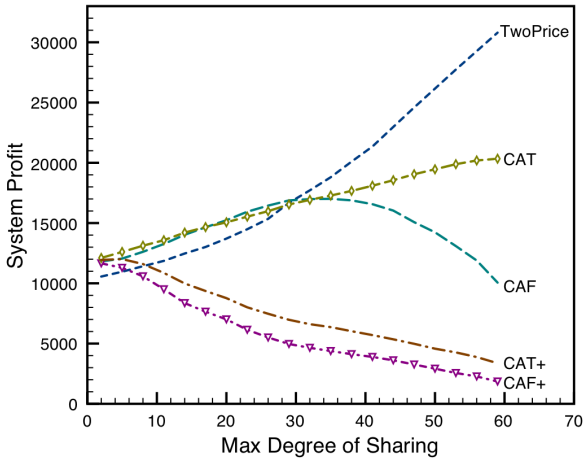
(a) System Capacity = 5000



(b) System Capacity = 10,000



(c) System Capacity = 15,000



(d) System Capacity = 20,000

Figure 8: The sequence of figures in 8(a) through 8(d) shows system profit as system capacity varies from 5000 to 20,000 in increments of 5000.

Table 5: Runtime performance averages for each algorithm on 50 workloads with 2000 queries

Random	GV	Two-price	CAF	CAF+	CAT	CAT+
0.92	2.003	3.72	7.088	12555.5	7.26	10091.2

To simulate strategizing users, we add an alternative bid to each client, which represents a lower bid than her valuation, and it is the product of her query valuation (original bid) and a lying factor. If a user's query shares many operators with other queries, she would strategize by bidding lower than her valuation thus lowering her payment and increasing her payoff. Therefore, if the ratio of *Static Fair Share/Total Load* is less than a certain threshold, the client will lie (i.e., submit the alternative bid) with a certain probability. We generated two workloads: a moderately lying workload and an aggressively lying workload. In the moderately lying workload, the threshold is set to 0.25, the probability of lying to 0.5, and the lying factor to 0.5, while in the aggressively lying workload, they are set to 0.35, 0.7 and 0.3 respectively.

Figure 9 shows the profit for three strategyproof mechanisms, CAF, CAT and Two-price, along with three different representations of the profit of CAR: CAR when no user lies, CAR-ML (CAR running the Moderate Lying workload) and CAR-AL (CAR running the Aggressive Lying workload). We see that when some users lie, the system profit decreases, motivating the need and desire of the system for a strategyproof mechanism. The profit of the three strategyproof mechanisms is dependable, while the profit from CAR is manipulable.

Finally, Table 6 simply summarizes the desirable characteristics of each mechanism alongside its experimental performance for various metrics like profit maximization, total user payoff, and rate of admission. Note that all mechanisms listed here are strategyproof. Admission Rate, Total User Payoff, and Profit are all in terms of relative performance as degree of sharing increases. For Profit, note that in the special case that degree of sharing is high and system capacity is almost as high as total system demand, the profit from CAF and CAT begins to dwindle and the profit from Two-price is actually highest.

3.6 SUMMARY

In this work, we applied techniques and principles from algorithmic game theory to a data streams query admission control problem. We introduced a new auction problem that,

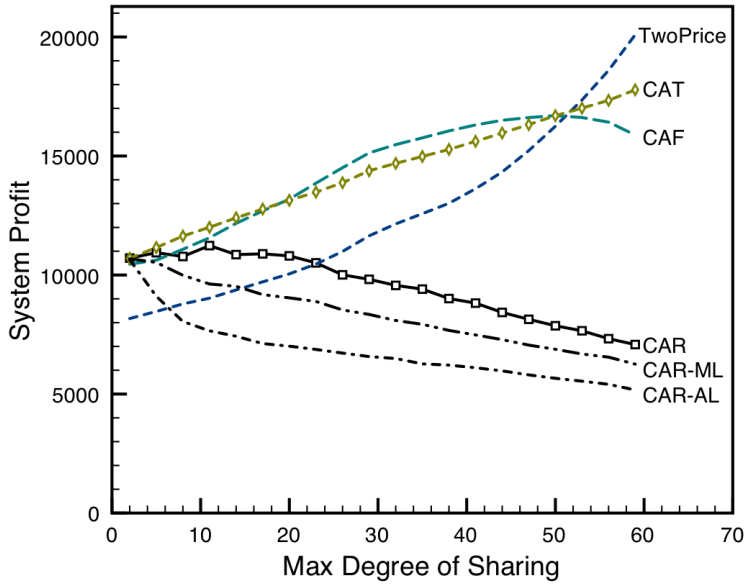


Figure 9: Profit of the three strategyproof mechanisms, (CAF, CAT, and Two-price), in comparison with the following different representations of the non-strategyproof mechanism CAR: CAR when no user lies, CAR-ML(CAR running the Moderate Lying workload) and CAR-AL(CAR running the Aggressive Lying workload). System capacity = 15,000.

Table 6: Properties of our proposed auction mechanisms.

Mechanism	Sybil Immune	Profit Guarantee	Admission Rate	Total User Payoff	Profit
CAF	×	×	High	Medium	High
CAF+	×	×	High	High	Low
CAT	✓	×	Medium	Medium	High
CAT+	×	×	Medium	High	Low
Two-price	×	✓	Low	Low	Medium

when posed abstractly, can be applied generally to naturally arising combinatorial settings outside of DSMSs. We introduced the notion of *sybil immunity* for auction mechanisms which can also be applied generally to any mechanism design setting. We proposed greedy and randomized auction mechanisms for this problem which are all *strategyproof*. We show the greedy approaches cannot give provable profit guarantees, and we also show that the randomized approach, for which we do give a provable profit guarantee, is not sybil immune. Our theoretical results pose a nice open question: can one find a mechanism for this problem that is strategyproof, sybil immune, *and* has a profit guarantee?

Our experimental results show that, generally speaking, CAT and CAF are the best mechanisms to use for profit maximization. However, if you have a high degree of operator sharing, and your system capacity is close to the total demand of the queries requesting service, then Two-price performs better for profit maximization. As expected, the greedy mechanisms (CAF, CAF+, CAT, and CAT+) provide better admission rate and payoff than Two-price. CAF+ and CAT+ are best for total user payoff, while CAF and CAF+ have the highest query admission rate as the degree of sharing increases. CAT, the one mechanism which is sybil immune, seems to offer the best overall tradeoff with respect to profit.

The data streams setting lends itself to natural and interesting variants on our model. For example, we might consider a more general and perhaps more realistic model where each query not only has a private valuation and a set of operators it wants executed, but it also has a specific interval of time during which it wants to be executed. The processing costs of each operator now represent costs per time unit, and the problem takes on a scheduling nature. Each time we choose to service a query we are committed to servicing it for the entire time interval specified by the query. And as before, we have a server of limited capacity. Our goal would again be to find a truthful mechanism that maximizes profit.

Another possible (and perhaps more realistic) variant is one where rather than having all queries and their corresponding information present up front, queries instead arrive over time. Hence a natural avenue for future work would include the design of *online* mechanisms for query admission.

And finally, we might consider the issue of energy consumption of the DSMS center Different levels of system operation incur different energy costs. This can be coupled with

the observation that it might be more profitable not to fully utilize the available capacity. Indeed, this is what our results suggest. Hence, an extension is to maximize profit while taking energy costs from system utilization into consideration.

4.0 INTERNET BOTTLENECK ROUTER CONGESTION

In the last chapter we discussed work that we completed toward our goal of using game theoretic models for problems in applied computer science. We now work toward both of our goals: demonstrating the utility of adaptive learning from evolutionary game theory and applying game theoretic techniques to real-world problems. Specifically, we study a problem from the field of networks.

4.1 INTRODUCTION

Ever since the first congestion control algorithms for TCP endpoints were introduced in [53], the important problem of congestion control at bottleneck routers on the Internet has garnered wide-spread attention. Several algorithms have been proposed for queue management and scheduling of packets in routers. Initially, such algorithms were designed under the assumption that all packets arriving at the routers come from TCP complying sources that produce packet flows with certain characteristics: all flows that become aware of congestion at the router (by seeing some of their packets dropped) will respond by reducing their transmission rates. However, TCP flows are not the only ones competing for available bandwidth or space in router queues. UDP flows behave in a completely different manner, tending to be more aggressive since they don't have TCP's built-in congestion control behavior. Moreover, the assumption that future users will continue using the current TCP protocol seems questionable. Since there is no central authority governing their behavior, as users compete for bandwidth, they may very well change the way they respond to congestion.

Studying congestion control from a game theoretic perspective is therefore natural. Us-

ing a variety of models, game theory has been used not only to find Nash equilibria (NE) when users are self-interested and routers employ existing methods, (e.g. FIFO with Drop-tail, or RED [42]) but also to design new router queuing policies, aimed at reaching good social outcomes in the presence of such users [46]. Such "good social outcomes" include the avoidance of congestion at routers, and thus avoidance of Internet congestion collapse, but also fairness of bandwidth sharing.

However, an approach commonly taken is to assume perfect information. Users are assumed to know the transmission rates of others and the congestion levels at the router, and use this information to compute a best response and optimize their utility. Even though such assumptions are standard, and even necessary, in the study of NE, they are not likely to be met in a setting like the Internet. Without such assumptions, can the equilibria be reached? Could there be a set of states, none of them necessarily a NE, such that the system gets trapped cycling among the states in the set? These are the questions we are aiming to answer in this work.

Using a simple yet general model of the game played by internet endpoints at internet bottleneck routers, we provide the first (to our knowledge) analysis of this problem using stochastic stability, a classical solution concept from the adaptive learning model of evolutionary game theory. Evolutionary game theory's adaptive learning setting is suited especially well for the game of internet endpoints competing for bottleneck router capacity. In traditional game theoretic settings, each player must assume all other players are perfectly rational, and must be fully informed of each other's actions and preferences. When players are internet endpoints, such requirements seem unreasonable and quite unlikely to be met. In our evolutionary setting, under adaptive learning's imitation play, players need only know what you would expect them to know: what they themselves experience in each round of play. They can then use simple heuristics to decide, based on the results of their recent play, what strategy to employ for the next round of play.

To study our problem in this adaptive learning setting, we use a new model proposed by Efraimidis and Tsavlidis [28] called the *window game*. This model is not only simple, but more general than previous models in which players are usually assumed to be TCP endpoints with specific loss recovery properties. In the window game, the endpoints are modeled such

that they can be thought of as using either TCP, UDP, or whatever transmission protocol they choose. There are n internet endpoints, each seeking to send an unlimited amount of traffic. But all endpoints encounter the same bottleneck router, which has capacity C. Each of the endpoints is a player that chooses a strategy: an integer-sized "window" between 0 and C. The window size can be thought of as the amount of the router's capacity being requested by the player, or the number of packets being sent by the player.

The amount of capacity that the router actually allocates to each player is then determined by the router's queuing policy and the specified window sizes (capacity requests) of the other users. We let each successful packet (each unit of successfully acquired capacity) equate to a profit to the player of 1, while each packet loss costs the player $g \geq 0$. Hence each player's utility or payoff can be expressed as:

$$(\text{number of successfully sent packets}) - g \cdot (\text{number of lost packets})$$

where the number of successful packets is equal to the amount of capacity granted to the player by the router, and the number of lost packets is the difference between the player's specified window size and the capacity that was actually granted to the player by the router.

We assume that this game is played repeatedly in rounds, in which every player chooses a strategy to play using imitation dynamics: sampling the outcomes of the rounds of play in its memory, and then imitating the strategy that served it best. However, with very small probability, each player fails to follow the imitation dynamics and chooses a strategy at random. Then, loosely speaking, the set of *stochastically stable states* represents the set of strategy profiles that have positive probability of being played in the long run, or, the states that the system eventually settles on. More details can be found in Section 4.2.1.

4.1.1 Our Results

We begin by analyzing the two currently most well-known and widely-used router queuing policies: FIFO with Droptail and RED (Random Early Detection) [42]. When Droptail is used, all incoming packets are simply dropped once the queue is full. We show that for any reasonable value of g, the only NE and the only stochastically stable state is the state

where all players send $\frac{g+1}{g}C\frac{n-1}{n^2}$ packets. This implies, for instance, that for a large number of flows and any value of $g \leq 1$ ($g = 1$ means that each player is hurt by each lost packet about the same as the amount they gain from each successful packet), the router is getting hit by roughly more than twice as much traffic as it has capacity. Next, we show that under RED queuing, in which the router starts dropping packets preemptively as soon as its buffer reaches a certain threshold $T < C$, things are a little better, but not by much. For reasonable values of g, there is a single NE, which constitutes also the single stochastically stable state, in which the congestion at the router is still significant.

Finally we study a queue policy proposed by Gao et al. [46], in which any overflow is compensated for by dropping the packets belonging to the most demanding flow. This policy was designed, in a idealized setting, to have a unique NE such that the router capacity is equally shared among all flows and overflow is avoided. They also studied a non-idealized setting in which in which flows do not have perfect information, and all sources are restricted to fixed-rate Poisson rates except one, which can be arbitrarily aggressive. In this setting, they succeed at a more modest goal: the source that can be arbitrarily aggressive should not outperform the best Poisson source by much. In this work we show that this policy can actually do even better. We show that even if flows have no information about one another, and all of them can arbitrarily adjust their window sizes (so no flow is restricted to a fixed rate), the system will still converge to the fair equilibrium under adaptive learning with imitation dynamics. Players with no information about the game other than what they themselves experience, making quick, simple decisions based on recent play, still converge on the NE in the long run.

Even though the stochastically stable states for the queuing policies we study turn out to coincide with the NE, what our results indicate is the following: even in the chaotic internet setting, where players have extremely limited information about the game and make instantaneous decisions, the NE will actually be reached. More specifically, for the all three router queue policies under consideration, we show that the system will not be cycling indefinitely between any other set of states; on the contrary, the unique NE in each case will be obtained.

4.1.2 Related Work

FIFO with Droptail is the traditional queue policy that has been employed widely in internet routers. As soon as the router queue is full, all subsequent incoming packets are dropped. For more information on Droptail and its variants, see [17]. RED [42] works similarly, but starts dropping packets with a certain probability as soon as the number of packets in the queue exceeds a threshold $T < C$. Both these policies punish all flows in a similar manner, regardless of whether they are "responsible" for causing the overflow or not. Specifically, the expected fraction of the demand of each flow that gets through the router is the same among all flows, those with moderate demands, and those with demands far exceeding their "fair share," contributing more to the overflow. The result of such a queue policy is that flows with large demand can use more router capacity at the expense of lower-demand flows.

There have been methods suggested for inhibiting such behavior. The Fair Queuing algorithm [25] ensures the max-min fairness criterion: using round-robin for selecting the outgoing packets, every flow can at least obtain its "fair share." Even though this is a fair scheme, it comes at the cost of efficiency. It requires separate buffers for each queue and a lot of book-keeping, making it unusable in practice. A method that achieves the same result without the high computational cost at the routers was suggested in [78]. This method however, cannot be used independently in each router, as it depends on receiving flow-specific information from other routers.

CHOKe [70], on the other hand, is a stateless queue management scheme, which can be implemented in a router independently from what other routers use. Every time a packet arrives to the queue, it is compared to $M \geq 1$ packets chosen uniformly at random from those currently in the queue and if it comes from the same source as any of them, then the two matching packets are dropped. There are both theoretical and experimental studies [82, 59] suggesting its effectiveness at preventing greedy (e.g. UDP) flows from strangling moderate flows. However, as the number of greedy flows varies, the number of packets chosen for comparison must also change in order to protect the more moderate flows from losing their fair share.

Gao et al [46] introduce a router queue management algorithm, which, unlike Fair Queu-

ing, does not require separate buffers for each flow, but, under some assumptions, achieves the same (fair) NE as max-min fairness. The main idea is to keep track of the "greediest" flow. Whenever there is an overflow, the algorithm drops only packets that belong to this flow. [1] The *Prince* algorithm described in [28] works in a similar manner. The algorithm in [46] was aiming to fulfill, among others, the following two objectives. First, in an idealized environment of full information, the profile corresponding to max-min fairness is the unique NE. Second, removing the full information setup but restricting all flows but one to being Poisson sources of fixed rates, the unrestricted flow cannot get a much better throughput than the best Poisson flow, no matter how it varies its sending rate.

There are several game theoretic results for congestion control. For a better introduction, we refer the reader to [76] and [56]. Akella et al. [3] study the equilibria of a TCP game, in which all flows use the Additive Increase Multiplicative Decrease (AIMD) algorithm. This is the method currently employed by TCP endpoints. The strategy sets consist of the possible values for the parameters of the algorithm. They show that even though the older TCP endpoint implementations can lead to efficient equilibria even with FIFO Droptail and RED router queue policies, this is no longer the case with newer implementations. They show that some measure of "network efficiency" can be established with a variant of CHOKe, assuming however that all flows are TCP. A lot of work has been devoted to game theoretic models in which all flows originate from Poisson sources and each source is allowed to vary the transmission rate [76, 26, 27]. The inefficiency of NE is studied, mainly in the case of a single bottleneck router, but also in more general networks [47]. Kesselman et al [56] consider a model in which the flows are explicitly deciding when to send new packets, instead of implicitly modifying their transmission rates.

With respect to adaptive learning and computer science: the game of users setting transmission rates for optimally receiving multimedia traffic is analyzed using an evolutionary game theoretic approach based on adaptive learning in [64]. And in [21], adaptive learning with imitation dynamics was used to analyze a load balancing game.

The Window game model we study here was first proposed in [28], where it was used

[1] Only in case that the overflow is greater than the number of packets of the greediest flow in the queue, will packets from other flows be dropped as well.

to find the NE in games between AIMD but also more general flows. We believe that this model is simple enough to allow interesting theoretical analysis, but still captures the essence of the game played between competing internet endpoints.

4.2 MODEL, NOTATION, AND BACKGROUND

To model internet endpoints competing for capacity at a bottleneck router, we use the window game of [28]. Let N be the set of players, $|N| = n$, with each player representing an internet endpoint. The strategy set for each player is the set of all possible window sizes, integer values between 0 and C, where C is the capacity of the bottleneck router. Let w_i be the window size requested by player i. Let $w = (w_1, w_2, \ldots, w_n)$, so w is a *strategy profile* vector or *solution* of the game. Let w_{-i} refer to the vector of all the strategies in w except w_i. Let $W = \sum_{i=1}^{n} w_i$ and let $W_{-i} = W - w_i$. The bottleneck router uses a (possibly randomized) queuing algorithm (like Droptail, RED, etc), to decide how many of each player's packets to keep, and how many to drop. Therefore the queuing policy maps each strategy profile w to a corresponding vector that indicates for each player i how many of its w_i packets are kept (in expectation), $keep_i$, and how many are dropped, $w_i - keep_i$. As described in the previous section, $g \geq 0$ is a real value that indicates how much detriment a lost packet causes to each player. Then for any $i = 1 \ldots n$, the payoff function of i is $\pi_i(w) = (keep_i) - g(w_i - keep_i)$.

A *best response* to w_{-i} for each player i is then $br_i(w_{-i}) = \arg\max_{w_i} \pi_i(w_i, w_{-i})$.

4.2.1 Adaptive Learning and Imitation Dynamics

We now more formally present the relevant aspects of evolutionary game theory's adaptive learning model [43, 87, 88], as well as the imitation dynamics of [54]. A related, more detailed summary can be found in Chapter 2, in which adaptive learning and imitation dynamics are applied to a load balancing game.

In the adaptive learning model with imitation dynamics, each of n players has a finite memory of their own actions and payoffs in the previous m rounds of play. After each round,

each player samples (uniformly at random) s of the m previous rounds of play, and then in the next round, plays the strategy (in our case, the window size) that yielded highest average payoff over the rounds that were sampled. In this way, the player is "imitating" the strategy that has served her well in the past.

These dynamics correspond to a Markov process P, where each state in the process is the history of the last m rounds of play. Each play history is comprised of m strategy profiles, and a state where all m strategy profiles are the same is called a *monomorphic* state.[2] The transition probabilities between states of the process are determined by the imitation dynamics described above. A *recurrent class* of a Markov process is a set of states such that there is zero probability of leaving the set once a state in the set has been reached, but positive probability of reaching any state in the set from any other state in the set. Josephson and Matros [54] prove the following about the process P.

Theorem 4.2.1 ([54])**.** *If $s/m \leq 1/2$, a subset of states is a recurrent class if and only if it is a singleton set containing a monomorphic state.*

If we now suppose that in each round, each player with probability $\epsilon > 0$ does not follow the imitation dynamics, but instead chooses a strategy at random, we have modified the Markov process so that there is always positive probability of eventually reaching any state from any other state. Therefore, there is a unique stationary distribution over the states in this modified process. We refer to this modified process as the *perturbed* Markov process, P^ϵ and the stationary distribution as μ^ϵ. The *stochastically stable states* are those states h in this modified process for which $\lim_{\epsilon \to 0} \mu^\epsilon(h) > 0$.

A *better reply* is a unilateral strategy deviation by a player that gives that player at least as high a payoff as the original strategy profile. I.e., x is a better reply for player i if $\pi_i(x, w_{-i}) \geq \pi_i(w)$. A *cusber set* or a set "closed under single better replies," is a set of strategy profiles such that any sequence of better replies, by any sequence of players, starting from any strategy profile in the set, always leads to another strategy profile that is also in the set. A *minimal cusber set* is a cusber set such that if any strategy profile is removed,

[2]For expository simplicity, if a monomorphic state has w as the strategy profile that fills its history, we will sometimes abuse notation and use w not just as the name of the strategy profile, but when the context is clear, as the name of the monomorphic state containing w.

the remaining set is no longer a cusber set.

Theorem 4.2.2 ([54]). *Under imitation dynamics, the profiles in the set of stochastically stable states are a minimal cusber set or a union of minimal cusber sets.*

Note that the following corollary is an immediate consequence of Theorem 4.2.2.

Corollary 4.2.3. *If a single strategy profile comprises the only minimal cusber set in a game, then that is the only strategy profile in the set of stochastically stable states under imitation dynamics.*

For a more complete background on stochastic stability and imitation dynamics, we refer the reader to [88, 54]. In what remains, we assume that $s/m \leq 1/2$.

4.3 DROPTAIL

FIFO queues with Droptail are widely used in Internet routers. While the queue has not reached its capacity, incoming packets are inserted in the end of the queue. As soon as the capacity is reached, any new incoming packets are dropped. We will start by describing the window game model of Droptail, then discuss the NE, and finally prove that there is a single stochastically stable state that corresponds to the unique NE.

Remember that for any profile w, we denote by W the total window size requested, i.e., $W = \sum_{i=1}^{N} w_i$. Under the Droptail routing policy, when $W > C$, the router chooses $W - C$ packets uniformly at random to be dropped. Therefore, for any player i with a window size w_i, the expected number of packets of i that will enter the queue is $w_i \cdot C/W$, while $w_i \cdot (1 - C/W)$ will be dropped. Of course, when $W \leq C$ then no packets will be dropped. This means that the expected payoff under Droptail for any player i can be expressed as

$$\pi_i^{DT}(w) = \begin{cases} w_i & \text{if } w_i \leq C - W_{-i} \\ w_i C/W - g w_i \,(1 - C/W) & \text{if } w_i > C - W_{-i} \end{cases} \tag{4.1}$$

We will refer to the first piece of the payoff function as $\pi_i^{DT_1}$ (the *linear piece*), the second piece as $\pi_i^{DT_2}$ (the *Droptail piece*), for obvious reasons. We note that when the total window

size equals the capacity, i.e., $w_i + W_{-i} = C$, then both pieces of the payoff function result in the same payoff. Therefore, for $W = C$ either of the two subcases can be used.

4.3.1 Droptail's Best Response

First we determine the best response function based on the above payoff function when Droptail is used, so that we can then characterize the NE of the game.

Theorem 4.3.1 (Best Response)**.** *The best response to player i can be expressed*

$$br_i(w_{-i}) = \begin{cases} C - W_{-i} & \text{if } W_{-i} \leq z_g \\ \sqrt{\frac{(g+1)CW_{-i}}{g}} - W_{-i} & \text{if } W_{-i} > z_g \end{cases} \tag{4.2}$$

where $z_g = Cg/(g+1)$ is the value such that $C - z_g = \sqrt{\frac{(g+1)Cz_g}{g}} - z_g$. [3]

Proof. We refer to the first piece of the best response function as br^{DT_1} and the second piece as br^{DT_2}. Note that $br^{DT_1}(W_{-i})$ maximizes $\pi_i^{DT_1}(w_i, W_{-i})$ within the range of its validity, and $br^{DT_2}(W_{-i})$ maximizes $\pi_i^{DT_2}(w_i, W_{-i})$. The difference $br^{DT_1}(W_{-i}) - br^{DT_2}(W_{-i})$, is 0 for $W_{-i} = z_g$, while it is positive for $W_{-i} < z_g$, and negative for $W_{-i} > z_g$. As noted after the definition of π_i^{DT}, for any $W_{-i}, 0 \leq W_{-i} \leq C$, $\pi_i^{DT_1}(C - W_{-i}, W_{-i}) = \pi_i^{DT_2}(C - W_{-i}, W_{-i})$. Also, for any $W_{-i} \geq 0$, $\pi_i^{DT_1}(br^{DT_1}(W_{-i}), W_{-i}) = \pi_i^{DT_1}(C - W_{-i}, W_{-i}) = \pi_i^{DT_2}(C - W_{-i}, W_{-i}) \leq \pi_i^{DT_2}(br^{DT_2}(W_{-i}), W_{-i})$.

However, when $W_{-i} < z_g$, $br^{DT_2}(W_{-i}) < br^{DT_1}(W_{-i}) = C - W_{-i}$, implying that if the second piece of the best response function was used (i.e., the one corresponding to br^{DT_2}), then the total window size would be less than C. In that case it is the $\pi_i^{DT_1}$ piece of the payoff function that applies, and that gets maximized at $C - W_{-i}$. Moreover, for any $w_i > br^{DT_2}(W_{-i})$, the function $\pi_i^{DT_2}(w_i, W_{-i})$ is decreasing on w_i. So, for any w_i such that $w_i \geq C - W_{-i}$, $\pi_i^{DT_2}(w_i, W_{-i}) \leq \pi_i^{DT_2}(C - W_{-i}, W_{-i}) = C - W_{-i}$.

[3] Here, the second piece is more formally expressed as $\min\{C, \sqrt{\frac{g+1}{g}CW_{-i}} - W_{-i}\}\}$, since, when $g < 1/3$, there are certain values of W_{-i} such that the value of the second piece function is larger than C. For ease of presentation, in what follows we will assume that $g \geq 1/3$. All statements can however be easily adjusted for the case that $g < 1/3$.

On the other hand, if $W_{-i} > z_g$, then $br^{DT_2}(W_{-i}) > br^{DT_1}(W_{-i}) = C - W_{-i}$, implying that $W_{-i} + br^{DT_2}(W_{-i}) > C$, and the maximum payoff $\pi_i^{DT_2}(br^{DT_2}(W_{-i}), W_{-i}) > \pi_i^{DT_1}(br^{DT_1}(W_{-i}), W_{-i})$ is obtained using the br^{DT_2} piece of the best response function. □

4.3.2 Droptail's NE

In this section we study Droptail's NE. The following lemma about symmetric NE under Droptail was given by Efraimidis and Tsavlidis in [28].

Lemma 4.3.2. *For $g \leq n-1$, the profile in which all players play window size d_g is the unique symmetric NE.*

In fact, $(d_g, \ldots, d_g)$ is the only NE of any kind (symmetric or not) for the case $g \leq n-1$. To show this, we start by making the observation that at any NE, all players are playing window sizes that correspond to a best response calculated using the same "piece" of our piecewise-defined best response function.

Lemma 4.3.3. *At any NE, either $W_{-i} \leq z_g$ for all i, or $W_{-i} \geq z_g$ for all i.*

Proof. Recall from the proof of Theorem 4.3.1 the definitions of the functions π_a, π_b, b_a, and b_b. By definition, at any NE, everyone is playing a best response. Assume by way of contradiction that there is some player j for whom $W_{-j} > z_g$ and another player k for whom $W_{-k} < z_g$. This implies that $w_j + W_{-j} = C$, while $w_k + W_{-k} > C$. But $w_j + W_{-j} = w_k + W_{-k} = W$, therefore no two such players j, k exist. □

We are now ready to examine the first of two cases.

Lemma 4.3.4. *When $W > C$, the solution where all players play window size d_g is the only NE.*

Proof. Let $\mathcal{A}$ be a NE such that $W > C$ and let $X \subseteq N$ be the set of the players whose window size in $\mathcal{A}$ is greater than zero.

Consider any two players $i, j \in X$. Let $W_{-ij} = W - w_i - w_j$. Since for all players k, $W_{-k} > z_g$ by assumption, we have $w_i = br_i(W_{-i}) = \sqrt{(g+1)C(w_j + W_{-ij})/g} - (w_j + W_{-ij})$

and $w_j = br_j(W_{-j}) = \sqrt{(g+1)C(w_i + W_{-ij})/g} - (w_i + W_{-ij})$. Subtracting the two equations yields

$$\sqrt{(g+1)C(w_j + W_{-ij})/g} - (w_j + W_{-ij}) = \sqrt{(g+1)C(w_i + W_{-ij})/g} - (w_i + W_{-ij}),$$

hence $w_i = w_j$. Let $x = |X|$. Since $w_i = w_j$ for every $i, j \in X$ and $w_k = 0$, for every $k \in N \setminus X$, we obtain that $w_i = \sqrt{\frac{g+1}{g}C(x-1)w_i} - (x-1)w_i$ which gives $w_i = \frac{(g+1)(x-1)C}{gx^2}$. Assume now that $N \setminus X \neq \emptyset$ and let $k \in N \setminus X$. $W_{-k} > z_g$ and $w_k = 0$, therefore $\sqrt{\frac{g+1}{g}CW_{-k}} - W_{-k} \leq 0$ if and only if $W_{-k} \geq \frac{g+1}{g}C$. But $W_{-k} = x\frac{(g+1)(x-1)C}{gx^2}$. Combining the last two statements implies $\frac{x-1}{x} \geq 1$ which is false. Therefore $N \setminus X = \emptyset$, and $x = n$, which means the only NE point at which $W > C$ is the one where every player plays d_g. □

The proof of Lemma 4.3.4 implies the following about the maximum value that the sum of the window sizes of all players may take.

Corollary 4.3.5. *When $g \leq n-1$, in any NE such that $W > C$, it will always be $W < (1 + \frac{1}{g})C$.*

We are now ready to prove our main theorem of this section.

Definition 4.3.6. Define d_g to be $\frac{g+1}{g}C\frac{n-1}{n^2}$.

Theorem 4.3.7. *If $g \leq n-1$, then the outcome in which each player's window size is d_g is the only NE.*

Proof. By Lemma 4.3.3, we know that either $W_{-i} > z_g$ for all i, or $W_{-i} \leq z_g$ for all i.

Case 1: $W_{-i} > z_g$ for all i. We know from Lemma 4.3.4 that $(d_g, \ldots, d_g)$ is the only possible NE in this case. The condition on g implied by $W_{-i} > z_g$ when each player plays d_g is precisely $g < n-1$.

Case 2: $W_{-i} \leq z_g$ for all i. $W_{-i} \leq z_g$ implies $w_i \geq C/(g+1)$ for all players i. This lowerbound on the window size of any player along with the property that any NE w in this case must have $W = C$, means that $n \leq g+1$. Hence $g \geq n-1$ at any NE. Since $g \leq n-1$ by assumption, we know that at any NE in this case, $g = n-1$. Plugging this value for g back into our lowerbound on w_i gives $w_i \geq C/n$. But when $g = n-1$, $z_g = (n-1)C/n$, so we also know $w_i \leq C/n$. Hence the only equilibrium is $(C/n, \ldots, C/n)$, and the value of d_g is precisely C/n when $g = n-1$. □

For completeness, we now discuss the case where $g > n-1$. In this case, there is actually a spectrum of NE. First we observe that $(d_g, \ldots, d_g)$ is not a NE if $g > n-1$.

Lemma 4.3.8. *The profile in which every player plays d_g is a NE only if $g \leq n-1$.*

Proof. Assume that $g > n-1$. This implies that $n \cdot d_g < C$, therefore the payoff of every player is given by the function $\pi_a(w_i, W_{-i}) = w_i$. But if $W = n \cdot d_g < C$, then any player i can increase her payoff by $C - W$, by increasing her window size by the same amount. Therefore $(d_g, \ldots, d_g)$ is not a NE. □

Theorem 4.3.9. *If $g > n-1$, then the only NE are those states where $W = C$. Specifically, w is a NE if and only if at w, $W = C$ and each player's window size falls in the range $\frac{C}{g+1}, \ldots, C - \frac{C}{g+1}$.*

Proof. Let w be a NE. By Lemma 4.3.3, we know that either $W_{-i} \leq z_g$ for all i or $W_{-i} \geq z_g$ for all i. Lemma 4.3.4 implies that when $W_{-i} > z_g$, the only possible NE is the profile $(d_g, \ldots, d_g)$. This profile, however, is not a NE when $g > n-1$ (by Lemma 4.3.8). Therefore, in any NE $W_{-i} \leq z_g$ for all $i \in N$. In this case, $br(W_{-i}) = C - W_{-i}$ for all i, and hence any NE must have the property that $W = C$. Moreover, for all $i \in N$, $W_{-i} \leq z_g$ and $w_i + W_{-i} = C$, therefore $w_i \geq C - z_g = \frac{C}{g+1}$. □

4.3.3 Droptail's Stochastically Stable States

In the following, we will assume that $g \leq n-1$, since the case where $g > n-1$ is of no practical relevance. We will now establish that the state $(d_g, \ldots, d_g)$ is the only SSS. Our proof uses the fact that any profile in a stochastically stable state is found in a minimal cusber set (Theorem 4.2.2), along with the fact that under the Droptail routing policy, the only minimal cusber set in our game is the NE profile itself. We first give three lemmas that allow us to establish the latter fact, by showing there is a better-reply path from any profile to the NE profile.

Lemma 4.3.10. *For any $W \geq C$, and any player i, such that $W_{-i} + d_g \geq C$, d_g is a better response for player i, if and only if $(d_g - w_i)\,(C(g+1)W_{-i} - g(d_g + W_{-i})W) \geq 0$.*

Proof. Consider any $W \geq C$, and some player i, such that $W_{-i} + d_g \geq C$. Player i has d_g as a better response if and only if $\pi_b(d_g, w_{-i}) \geq \pi_b(w_i, w_{-i})$, or

$$\begin{aligned}
&\frac{d_g}{d_g + W_{-i}} C(g+1) - g d_g \geq \frac{w_i}{W} C(g+1) - g w_i \\
\Leftrightarrow \quad & C(g+1)\left(d_g(W_{-i} + w_i) - w_i(d_g + W_{-i})\right) - g(d_g + W_{-i}) W (d_g - w_i) \geq 0 \\
\Leftrightarrow \quad & C(g+1) W_{-i}(d_g - w_i) - g(d_g + W_{-i}) W (d_g - w_i) \geq 0 \\
\Leftrightarrow \quad & (d_g - w_i)\left(C(g+1) W_{-i} - g(d_g + W_{-i}) W\right) \geq 0.
\end{aligned}$$

□

Lemma 4.3.11. *Let $w \neq (d_g \ldots, d_g)$, $W \geq C$. Within at most two better replies, a profile w' can be reached, such that for any k with $w_k = d_g$, $w'_k = d_g$, and there is some player i, such that $w_i \neq d_g$ and $w'_i = d_g$. Moreover $W' \geq C$.*

Proof. We will show that either there is immediately some player i with $w_i \neq d_g$ that has d_g as a better response, or after just one more better response move by another player, such a player i will exist. Thus we will show that after at most two better response moves, we will be in a profile w' where some player i who did not play d_g in profile w is playing d_g in w'. (We will do this while ensuring that any player that played d_g in w, still plays d_g in w'.)

First consider the case that $W \leq n \cdot d_g$. Consider some player i such that $w_i \leq W/n$ and $w_i < d_g$. (Note that such a player must exist.) Lemma 4.3.10 implies that d_g is a better reply to i if and only if $C(g+1)W_{-i} - g(d_g + W_{-i})W \geq 0$, or equivalently,

$$(W - w_i)(C(g+1) - gW) \geq gW d_g. \tag{4.3}$$

Now, by assumption $W \leq n d_g = \frac{g+1}{g}\frac{n-1}{n} C < \frac{g+1}{g} C$, and thus the last inequality (4.3) is equivalent to

$$w_i \leq W - W \frac{g d_g}{C(g+1) - gW} = W - W \frac{d_g}{d_g \frac{n^2}{n-1} - W}.$$

However, since $W \leq n \cdot d_g$, we have

$$\begin{aligned} W - W\frac{d_g}{d_g\frac{n^2}{n-1} - W} &\geq W - W\frac{d_g}{d_g\frac{n^2}{n-1} - nd_g} \\ &= W - W\frac{n-1}{n^2 - n(n-1)} = W - W\frac{n-1}{n} = \frac{W}{n} \end{aligned}$$

Therefore, $w_i \leq \frac{W}{n} \leq W - W\frac{d_g}{d_g\frac{n^2}{n-1} - W}$, meaning d_g is a better response for player i.

In the case that $W > n \cdot d_g$, we have two subcases. (Note that in this case there is always some player k, such that $w_k \geq W/n$ and $w_k > d_g$.)

Case 1: There is a player i is such that $w_i \geq W/n$, $w_i > d_g$, and $W_{-i} + d_g \geq C$. Then Lemma 4.3.10 implies that d_g is a better response for player i if and only if $C(g+1)W_{-i} - g(d_g + W_{-i})W \leq 0$, i.e.,

$$\frac{W}{C} \geq \frac{g+1}{g}\frac{W_{-i}}{d_g + W_{-i}}. \tag{4.4}$$

If again $C(g+1) - gW > 0$, then the claim holds similarly with the case that $W \leq nd_g$. Otherwise, $gW \geq C(g+1)$ means $\frac{W}{C} \geq \frac{g+1}{g} \geq \frac{g+1}{g}\frac{W_{-i}}{W_{-i}+d_g}$.

Case 2: For all j such that $w_j \geq W/n$ and $w_j > d_g$, $W_{-j} + d_g < C$. Consider one such j and note that $W_{-j} < C - d_g < nd_g$ (since $g < n-1$). We will show that playing $nd_g - W_{-j}$ is a better response for j. But then the new total window size W' is equal to $nd_g > C$ and thus, (according to the case that $W \leq nd_g$) there is a player i, with $w_i \leq W'/n, w_i < d_g$ that has d_g as a better response.

To see that $nd_g - W_{-j}$ is a better response for j, note that the derivative $\frac{\partial \pi_b(w_j, w_{-j})}{\partial w_j} = \frac{\partial}{\partial w_j}\left(\frac{w_j}{W_{-j}+w_j}C(g+1) - gw_j\right) = \frac{W_{-j}}{(W_{-j}+w_j)^2}C(g+1) - g$ has a unique positive root (at $x_0 = br_b(w_{-j})$) and is negative for all $x > x_0$. Moreover, its value when player j plays $nd_g - W_{-j}$ is $\frac{W_{-j}}{n^2 d_g^2}C(g+1) - g < \frac{(n-1)d_g}{n^2 d_g^2}C(g+1) - g = \frac{(n-1)C(g+1)}{\frac{g+1}{g}(n-1)C} - g = 0$.

Therefore, since $w_j = W - W_{-j} > nd_g - W_{-j}$, we easily obtain that $\pi_i(w_j, W_{-j}) < \pi_j(nd_g - W_{-j}, W_{-j})$. □

Lemma 4.3.12. *For any $w \neq (d_g, \ldots, d_g)$, there is a finite sequence of better replies that lead to the profile $(d_g, \ldots, d_g)$.*

Proof. We note first that if $W < C$, then for any player i, playing $C - W_{-i}$ is a better response than w_i. Hence we will assume that $W \geq C$. Note that applying Lemma 4.3.11 to $w \neq d_g$, we will obtain some w' such that still $W' \geq C$. Therefore, by simply invoking Lemma 4.3.11 at most n times, we can see that there is a path of (in total) at most $2n + 1$ better response moves from w to the profile $(d_g, \ldots, d_g)$. □

We are now ready to proof our main theorem of this section.

Theorem 4.3.13. *For $g < n - 1$, the state in which every player plays d_g is the unique stochastically stable state.*

Proof. First of all, note that d_g is the unique better response to $W_{-i} = (n-1)d_g$. Therefore the profile $a = (d_g, \ldots, d_g)$ is a minimal cusber set. Moreover, by Lemma 4.3.12, there is a better response path from any $w \neq a$ to a. Therefore any other cusber set would have to contain a, which implies there is no other minimal cusber set. Hence, by Corollary 4.2.3, a is the only state that is stochastically stable. □

4.4 RED (RANDOM EARLY DETECTION)

RED (Random Early Detection) [42] was meant to keep the average queue size low. It works similarly to Droptail, but starts dropping packets before the queue is full. When the total load at the router exceeds a system-defined minimum threshold T, the router begins dropping each new arriving packet with probability proportional to the load. After total load exceeds a system-defined maximum threshold, the packets are dropped with probability 1. (Note that when the maximum threshold is set to C, then once capacity is reached, RED behaves exactly like Droptail.)

To simplify our study, we will assume that the maximum threshold is C, but we will leave the minimum threshold T as a free parameter. We then must model the RED protocol as a window game. In what follows we use the term *kept* to refer to the event of a packet not being dropped.

Assume that the current load at the queue is $L \geq T$. Then, according to RED, each new arriving packet will be dropped with probability $\frac{L-T}{C-T}$. Assume that when W packets arrive sequentially, the expected number of them that are kept is x. In contrast to this sequential process where packets arrive one by one, using the window game we assume that given a strategy profile w, all W packets arrive at the same time. Hence each packet will be kept with probability $\frac{x}{W}$ (x packets are chosen uniformly at random). If $W \leq T$, then all packets are admitted.

Lemma 4.4.1. *Assume that RED is used and let w be a strategy profile such that $W > T$.*

i)If $W \geq W_C$, where $W_C = (C-T)H_{C-T} + T$ then the queue size reaches C.

ii) If $T \leq W < W_C$, then $T + \tilde{k}_W$ packets are kept in expectation, where $\tilde{k}_W \approx (C - T)\left(1 - e^{-\frac{W-T}{C-T}}\right)$. (So the probability for any packet to be kept is $\frac{\tilde{k}_W+T}{W}$.)

Proof. The proof uses the solution to the well-known coupon collector problem. We consider the case that W packets arrive sequentially. Consider the moment at which the queue size becomes $T + i - 1$, for some $i, 1 \leq i \leq C - T$. Let X_i be a random variable that represents the number of packets that arrive at the system until the queue size reaches $T + i$ (i.e., $X_i - 1$ is the number of packets that arrive to the router and get dropped until one is kept). According to the description of RED, when $T + i - 1$ packets are already in the queue, the probability that a newly arriving packet is dropped is $\frac{i-1}{C-T}$. This implies that

$$\mathbf{E}[X_i] = \frac{C-T}{C-T-i+1}.$$

Let H_j be the jth harmonic number.

i) $W_C = T + \mathbf{E}\left[\sum_{i=1}^{C-T} X_i\right] = T + \sum_{i=1}^{C-T} \frac{C-T}{C-T-i+1} = (C-T)H_{C-T} + T.$

ii) The total number of packets $\tilde{k}_W$ that are kept, out of the total of W that arrive, is given as the maximum k, such that $T + \mathbf{E}\left[\sum_{i=1}^{k} X_i\right] \leq W$, or equivalently,

$$\sum_{i=1}^{k} \frac{C-T}{C-T-i+1} \leq W - T.$$

In other words, we need the maximum k such that $(C-T)\left(H_{C-T} - H_{C-T-k}\right) \leq W - T$. Approximating H_j with $\ln j$ we get: $\ln(C-T-k) \geq \ln(C-T) - \frac{W-T}{C-T}$ which gives $\tilde{k}_W \approx (C-T)\left(1 - e^{-\frac{W-T}{C-T}}\right)$. □

In order to simplify our presentation (and to allow clean formulation of a best response function), we will approximate $\tilde{k}_W$ by $k_W = \frac{W-T}{H_{C-T}}$. Note that k_W is also a continuous function, while $k_T = 0$ and $k_{W_C} = C - T$; therefore, when W equals T (respectively, W_C), the total number of packets kept and sent through the queue and sent is T (respectively, C), in accordance with Lemma 4.4.1. The payoff function of flow i is now expressed as:

$$\pi_i^{RED}(w) = \begin{cases} w_i & \text{if } W \leq T \\ w_i(k_W + T)/W - gw_i\,(1 - (k_W + T)/W) & \text{if } T < W \leq W_C \\ w_i C/W - gw_i\,(1 - C/W) & \text{if } W > W_C \end{cases}$$

We will refer to the first piece of the payoff function as $\pi_i^{RED_1}$ (the *linear piece*), the second piece as $\pi_i^{RED_2}$ (the *RED piece*), and the third piece as $\pi_i^{RED_3}$ (the *Droptail piece*), for obvious reasons. We will refer to the condition on W corresponding to each piece as the *range of validity* for that piece.

4.4.1 RED's Best Response

From this payoff, we can determine that RED has three different best response functions, one for each range of possible g values. We begin by noting the following facts. (Along the way we also define the terms, $br_i^{RED_2}$, $br_i^{RED_3}$, α_g, β_g, and γ_g.)

1. $\pi_i^{RED_1}$ is maximized (within its range of validity) at the right-most point, when $w_i = T - W_{-i}$.
2. If $g < 1/(H_{C-T}-1)$, then $\pi_i^{RED_2}$ is always increasing, and therefore its maximum is at the right-most point in its range of validity, when $w_i = W_C - W_{-i} = (C-T)H_{C-T} + T - W_{-i}$.
3. If $g \geq 1/(H_{C-T} - 1)$, then $\pi_i^{RED_2}$ is maximized when

$$w_i = br_i^{RED_2} = \sqrt{\frac{(g+1)(H_{C-T} - 1)TW_{-i}}{g(H_{C-T} - 1) - 1}} - W_{-i}$$

(by basic calculus), and this best response is valid for $\pi_i^{RED_2}$ only when

$$\alpha_g = \frac{T(gH_{C-T} - g - 1)}{(g+1)(H_{C-T} - 1)} < W_{-i} \leq \beta_g = \frac{((C-T)H_{C-T} + T)^2(g(H_{C-T} - 1) - 1)}{T(g+1)(H_{C-T} - 1)}$$

since we must have $T < W \leq W_C = (C - T)H_{C-T} + T$.

4. $\pi_i^{RED_3}$ is maximized when

$$w_i = br_i^{RED_3} = \sqrt{\frac{(g+1)CW_{-i}}{g}} - W_{-i}$$

(as we saw in Section 4.3), and this best response is valid for $\pi_i^{RED_3}$ only when

$$W_{-i} > \gamma_g = \frac{g((C-T)H_{C-T}+T)^2}{(g+1)C}$$

since we must have $W > W_C = (C-T)H_{C-T} + T$.

Hence, the value of w_i that maximizes the overall payoff function, i.e., the overall best response window size for player i, depends on the value of g (and of course, W_{-i}). It turns out the RED payoff function has three different best response functions, and which one applies depends on the value of g. The corresponding range of values of g for each of the three possible best response functions for RED is shown in Figure 10.

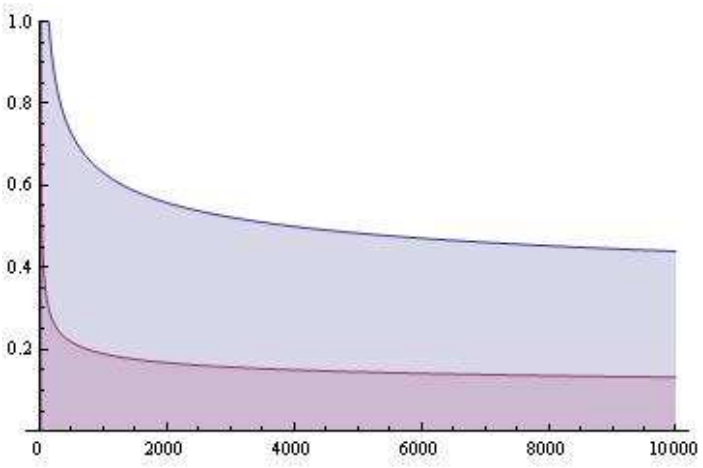


Figure 10: Here, g is shown as a function of C and the threshold T is set to $0.7C$. Each of the three best response functions corresponds to a region shown here. The lower range (darkest) is $g \leq 1/(H_{C-T} - 1)$, the middle range (lightly shaded) is $1/(H_{C-T} - 1) < g < C/((C-T)(H_{C-T} - 1))$, and the upper range (unshaded) is $g \geq C/((C-T)(H_{C-T} - 1))$.

We begin with the best response function for the lower-ranged values of g.

Theorem 4.4.2. *If $g \leq 1/(H_{C-T} - 1)$ then the RED best response function is*

$$br_i^{RED}(w_{-i}) = \begin{cases} (C-T)H_{C-T} + T - W_{-i} & \text{if } W_{-i} \leq \gamma_g \\ \max\left\{br_i^{RED_3}, 0\right\} & \text{if } W_{-i} > \gamma_g \end{cases} \tag{4.5}$$

Proof. For such small values of g, the linear piece of the RED payoff function always returns a lower payoff than the RED piece. And for such small values of g, $\pi_i^{RED_2}$ is strictly increasing in the range of its validity. So payoff is increasing with player i's window size up to the point where $w_i = (C-T)H_{C-T} + T - W_{-i}$. If $W_{-i} \leq \gamma_g$, then $br_i^{RED_3}$ is to outside of the range of validity for $\pi_i^{RED_3}$, in the negative direction. So at $(C-T)H_{C-T} + T - W_{-i}$, $\pi_i^{RED_3}$ has already begun decreasing. Therefore $(C-T)H_{C-T} + T - W_{-i}$ is the overall point of payoff maximization. If $W_{-i} > \gamma_g$, then $br_i^{RED_3}$ is within the range of validity for $\pi_i^{RED_3}$, so at $(C-T)H_{C-T} + T - W_{-i}$, payoff increases until $w_i = br_i^{RED_3}$. Hence $br_i^{RED_3}$ is the overall point of payoff maximization. □

The middle-ranged values of g yield the following best response function. Note that when $\beta_g < \gamma_g$, $g < C/((C-T)(H_{C-T} - 1))$.

Theorem 4.4.3. *If $1/(H_{C-T}-1) < g < C/((C-T)(H_{C-T}-1))$ then the RED best response function is*

$$br_i^{RED}(w_{-i}) = \begin{cases} T - W_{-i} & \text{if } W_{-i} \leq \alpha_g \\ \max\left\{br_i^{RED_2}, 0\right\} & \text{if } \alpha_g < W_{-i} \leq \beta_g \\ (C-T)H_{C-T} + T - W_{-i} & \text{if } \beta_g < W_{-i} \leq \gamma_g \\ \max\left\{br_i^{RED_3}, 0\right\} & \text{if } W_{-i} > \gamma_g \end{cases} \tag{4.6}$$

Proof. When g is in this middle range, there is a gap between values of W_{-i} for which the maximization point of $\pi_i^{RED_2}$, $br_i^{RED_2}$, is in its validity range and values for which the maximization point of $\pi_i^{RED_3}$, $br_i^{RED_3}$ is in its validity range. Since $br_i^{RED_3}$ is on the negative side (outside) of the validity border between the two pieces of the payoff function, and $br_i^{RED_2}$ is on the positive side (outside) of the validity border between the two pieces of the payoff function, the actual point of payoff maximization is the exact border point itself, $(C-T)H_{C-T} + T - W_{-i}$, which is reflected as the third piece of the best response function above. When $W_{-i} \leq \alpha_g$, the point $br_i^{RED_2}$ is on the negative side of its validity range, so

$\pi_i^{RED_2}$ is decreasing from the point $T - W_{-i}$ onward. Therefore $T - W_{-i}$ is the point of overall maximization. □

And finally, the upper-ranged values of g yield the following best response function. Note that when $\beta_g \geq \gamma_g$, $g \geq C/((C-T)(H_{C-T}-1))$.

Theorem 4.4.4. *If $g \geq C/((C-T)(H_{C-T}-1))$ then the RED best response function is*

$$br_i^{RED}(w_{-i}) = \begin{cases} T - W_{-i} & \text{if } W_{-i} \leq \alpha_g \\ \max\left\{br_i^{RED_2}, 0\right\} & \text{if } \alpha_g < W_{-i} \end{cases} \tag{4.7}$$

Proof. In this case, there is a range of values of W_{-i} where both $br_i^{RED_2}$ and $br_i^{RED_3}$ would be valid. However, in this case, starting at $W_{-i} = \gamma_g$, when the range of validity for $br_i^{RED_3}$ begins, $br_i^{RED_3}$ is always at a *non-positive* window size. Since from that point of payoff maximization of $\pi_i^{RED_3}$, as the player's window size grows, her payoff only decreases, playing a window size of 0 will always yield at least as high a payoff with respect to $\pi_i^{RED_3}$ as playing any positive window size. Hence it is still safe to simply use the $br_i^{RED_2}$ for any $\gamma_g \leq W_{-i} < \beta_g$, where β_g is (as defined above) the point at which the range of validity for $br_i^{RED_2}$ ends. For values of $W_{-i} \geq \beta_g$, $br_i^{RED_2}$ always returns a negative window size, therefore 0 will actually always be returned. □

4.4.2 RED's Nash Equilibria

As the best response function changes according to g, so do the NE of RED vary according to the value of g. We note here that, just as in Droptail, we assume that $g \leq n-1$, since any value greater than that is of no practical interest. We also use the following notation to refer the three ranges of values for g: $R_g^{low} = \left(0, \frac{1}{H_{C-T}-1}\right]$, $R_g^{mid} = \left[\frac{1}{H_{C-T}-1}, \frac{C}{(C-T)(H_{C-T}-1)}\right)$, and $R_g^{hi} = \left[\frac{C}{(C-T)(H_{C-T}-1)}, n-1\right]$.

For the lower-ranged values of g, the RED best response function is similar to Droptail's best response, both of them using $br_i^{RED_3}$. Hence, the proof of the following theorem about the NE for the lower-ranged values of g is analogous to that of Theorem 4.3.7 in Section 4.3.2 and we omit it here. Recall from Definition 4.3.6 the definition of d_g.

Theorem 4.4.5. *If $g \leq 1/(H_{C-T}-1)$ then the only NE is when $w_i = d_g$ for all i.*

We now establish the NE for the middle-ranged values of g.

Definition 4.4.6 (RED convergence point)**.** Define $r_g = \frac{T(g+1)(H_{C-T}-1)(n-1)}{(gH_{C-T}-g-1)n^2}$.

Theorem 4.4.7. *Recall from Definitions 4.3.6 and 4.4.6 the definitions of d_g and r_g. If $g \in R_g^{mid}$ then the only NE are:*

1. *$w_i = r_g$ for all i (for $\frac{T+H_{C-T}(Cn-T)}{H_{C-T}^2(Cn-Tn)+H_{C-T}(Tn-Cn+T)-T} \leq g$)*
2. *any solution such that $\sum_i w_i = (C-T)H_{C-T} + T$ (for $\frac{(n-1)C}{((C-T)H_{C-T}+T)n-(n-1)C} \leq g < \frac{T+H_{C-T}(Cn-T)}{H_{C-T}^2(Cn-Tn)+H_{C-T}(Tn-Cn+T)-T}$), or*
3. *$w_i = d_g$ for all i (for $g < \frac{(n-1)C}{((C-T)H_{C-T}+T)n-(n-1)C}$).*

Proof. As in the Droptail section, we know that at any NE, all players are using the same best response function, i.e., for all players i, the same condition on W_{-i} is satisfied.

First we can observe that g would be out of range for the $W_{-i} \leq \alpha_g$ case as follows. In this case we know that at any NE, $w_i = T - W_{-i} \geq T - \alpha_g$ for any i. Plugging in for α_g we see that $w_i \geq \frac{H_{C-T}T}{(H_{C-T}-1)(g+1)}$. Since $nw_i = T$, we can plug in for w_i and see that $g \geq nH_{C-T}/(H_{C-T}-1) - 1 \geq n-1$, much larger the upperbound on g for which this best response function even applies. Hence there can't be NE where all players are using the first piece of the best response function.

We then observe that in pieces 2 and 4 (the non-linear pieces) of the best response function, just as in Droptail, any NE must be symmetric. To see this for piece 2, refer to the proof of Theorem 4.4.8. The same observations can be carried out with piece 4 of the best response function.

The calculation of r_g can also be found in the proof of Theorem 4.4.8. We now calculate the more specific conditions on g under which the second piece of the best response function applies, i.e., the conditions under which $(r_g,...,r_g)$ is a NE. $W_{-i} > \alpha_g$ means $(n-1)r_g \geq T(gH_{C-T}-g-1)/((g+1)(H_{C-T}-1))$ at any SNE. That means $(n-1)(g+1)(H-1) \geq (gH_{C-T}-g-1)n$, which implies $g \leq nH_{C-T}/(H_{C-T}-1) - 1$. Note that this holds for any reasonable g and more importantly for all g in the range of consideration in the statement of the theorem.

$W_{-i} \leq \beta_g$ means

$$(n-1)r_g \leq \frac{((C-T)H_{C-T}+T)^2(g(H_{C-T}-1)-1)}{(T(g+1)(H_{C-T}-1))}$$

which means

$$T(g+1)(H_{C-T}-1)(n-1) \leq ((C-T)H_{C-T}+T)(g(H_{C-T}-1)-1)n$$

and hence

$$g \geq \frac{(CH_{C-T}n+T-TH_{C-T})}{(CH_{C-T}^2n+TH_{C-T}n+TH_{C-T}-CH_{C-T}n-TH_{C-T}^2n-T)}$$
$$= \frac{(T+H_{C-T}(Cn-T))}{(H_{C-T}^2(Cn-Tn)+H_{C-T}(Tn-Cn+T)-T)}.$$

Note this sometimes holds for g in the range under consideration in the statement of the theorem (depending on the values of n, C, and T).

The conditions on g for the NE to be where the players are all playing d_g can be similarly derived as follows: $W_{-i} > \gamma_g$ means $(n-1)d_g > g((C-T)H_{C-T}+T)/((g+1)C)$, which means $(n-1)(g+1)C > g((C-T)H_{C-T}+T)n$, and hence

$$g < \frac{(n-1)C}{((C-T)H_{C-T}+T)n-(n-1)C}.$$

Again note that this sometimes holds for g in the range under consideration in the statement of the theorem (depending again on values of n, C, and T.)

Finally, we consider the potential NE where all players are using the third piece of the best response function, i.e., when for all i, $w_i = (C-T)H_{C-T}+T-W_{-i}$. The conditions under which the third piece of the best response function applies are $W_{-i} > \beta_g$ and $W_{-i} \leq \gamma_g$. This means

$$w_i \geq (C-T)H_{C-T}+T-\gamma_g = (C_T)H_{C-T}+T-\frac{g((C-T)H_{C-T}+T)^2}{(g+1)C}.$$

Using this along with the SNE property that $nw_i = (C-T)H_{C-T}+T$ we can say

$$\begin{aligned} n &\leq \frac{(C-T)H_{C-T}+T}{(C-T)H_{C-T}+T-\frac{g((C-T)H_{C-T}+T)^2}{(g+1)C}} = \frac{1}{1-\frac{g((C-T)H_{C-T}+T)}{(g+1)C}} \\ &= \frac{(g+1)C}{(g+1)C-g((C-T)H_{C-T}+T)}. \end{aligned}$$

Isolating g gives us the stated lowerbound on g for this case. The upperbound can be computed similarly. □

And finally we establish the NE for the upper-ranged values of g.

Theorem 4.4.8. *If $g \in R_g^{hi}$, then there is a unique NE, such that $w_i = r_g$, for all i.*

Proof. We first observe that in the case where $br_i^{RED_2}$ (the RED piece of the best response function) applies, just as in Droptail, any NE must be symmetric. To see this, say that at some NE we are in the case where $\alpha_g < W_{-i} \leq \beta_g$ for all players i. Then for any two players j and k, we know that

$$w_j = \sqrt{\frac{(g+1)(H_{C-T}-1)TW_{-j}}{g(H_{C-T}-1)-1}} - W_{-j}$$

and

$$w_k = \sqrt{\frac{(g+1)(H_{C-T}-1)TW_{-k}}{g(H_{C-T}-1)-1}} - W_{-k}.$$

Subtracting the two equations gives

$$w_j - w_k = \sqrt{\frac{(g+1)(H_{C-T}-1)T}{g(H_{C-T}-1)-1}} \left(\sqrt{W_{-j}} - \sqrt{W_{-k}}\right) - W_{-j} + W_{-k}$$

which implies $W_{-j} = W_{-k}$.

Now that we've established the only NE on $br_i^{RED_2}$ of the function are SNE, the calculation of r_g comes from taking the RED piece of the best response function and checking what each player's window size must be if all players are playing that best response. In other words, solving the following equation for w_i:

$$w_i = \sqrt{\frac{T(g+1)(H_{C-T}-1)}{gH_{C-T}-g-1}(n-1)w_i} - (n-1)w_i.$$

We now calculate the more specific conditions on g under which the second piece of the best response function applies, i.e., the conditions under which $(r_g,...,r_g)$ is a NE. $W_{-i} > \alpha_g$ means $(n-1)r_g \geq T(gH_{C-T}-g-1)/((g+1)(H_{C-T}-1))$ at any SNE. This last inequality indeed implies that $g \leq nH_{C-T}/(H_{C-T}-1)-1$. □

4.4.3 RED's Stochastically Stable States

In this section we show that in the case where $g \in R_g^{hi}$ the only NE $(r_g, \ldots, r_g)$ is also the only stochastically stable state. We will only discuss here the case where g is in this upper range because it is the most practically relevant range of values for g. (In practice $T = \lambda C$, for some constant λ, so $\frac{C}{(C-T)(H_{C-T}-1)}$ is a decreasing function on C, tending to 0 as C grows large.) Our proof is similar to that for Droptail in Section 4.3.3, using the fact that any profile in a stochastically stable state is found in a minimal cusber set (Theorem 4.2.2), along with the fact that under the RED routing, the only minimal cusber set in our game is the NE state itself.

We begin with a sequence of Lemmas that allows us to establish the fact that there is a better-reply path from any profile to the profile $(r_g, \ldots, r_g)$.

Lemma 4.4.9. *For any $W \geq T$, and any player i, such that $W_{-i} + r_g \geq T$, r_g is a better response for player i, if and only if*

$$(r_g - w_i)((g+1)T(H_{C-T}-1))W_{-i} - (gH_{C-T} - g - 1)(W_{-i} + r_g)W) \geq 0$$

Proof. Consider any $W \geq T$, and some player i, such that $W_{-i} + r_g \geq T$. Player i has r_g as a better response if and only if $\pi^{RED2}(r_g, w_{-i}) \geq \pi^{RED2}(w_i, w_{-i})$, or

$$\begin{aligned}
& r_g \frac{\frac{W_{-i}+r_g-T}{H_{C-T}} + T}{W_{-i} + r_g}(g+1) - gr_g \geq w_i \frac{\frac{W-T}{H_{C-T}} + T}{W}(g+1) - gw_i \\
\Leftrightarrow\ & r_g(W_{-i} + r_g + T(H_{C-T}-1))W(g+1) - gH_{C-T}r_g(W_{-i}+r_g)W \\
& \geq w_i(W + T(H_{C-T}-1))(W_{-i}+r_g)(g+1) - gH_{C-T}w_i(W_{-i}+r_g)W \\
\Leftrightarrow\ & (r_g - w_i)(g+1)(W_{-i}+r_g)W - (r_g - w_i)gH_{C-T}(W_{-i}+r_g)W \\
& + r_g(g+1)(W_{-i}+w_i)T(H_{C-T}-1)) - w_i(g+1)(W_{-i}+r_g)T(H_{C-T}-1)) \geq 0 \\
\Leftrightarrow\ & (r_g - w_i)(g+1-gH_{C-T})(W_{-i}+r_g)W + (r_g - w_i)(g+1)T(H_{C-T}-1))W_{-i} \geq 0 \\
\Leftrightarrow\ & (r_g - w_i)((g+1)T(H_{C-T}-1))W_{-i} - (gH_{C-T} - g - 1)(W_{-i}+r_g)W) \geq 0
\end{aligned}$$

□

Lemma 4.4.10. *Let $w \neq (r_g \ldots, r_g)$, $W \geq T$. Within at most two better replies, a profile w' can be reached, such that for any k with $w_k = r_g$, $w'_k = r_g$, and there is some player i, such that $w_i \neq r_g$ and $w'_i = r_g$. Moreover $W' \geq T$.*

Proof. We first consider the case where $W \leq nr_g$. Note there must be some player i such that $w_i \leq W/n$ and $w_i < r_g$ (since not all players play r_g by assumption). Lemma 4.4.9 implies that r_g is a better reply to i if and only if

$$((g+1)T(H_{C-T}-1) - (gH_{C-T}-g-1)W)\,W_{-i} \geq Wr_g(gH_{C-T}-g-1) \tag{4.8}$$

Note that $W \leq nr_g = \frac{(g+1)T(H_{C-T}-1)}{gH_{C-T}-g-1}\frac{n-1}{n} < \frac{(g+1)T(H_{C-T}-1)}{gH_{C-T}-g-1}$. Hence we can use the fact that $W_{-i} = W - w_i$ to rewrite inequality 4.8 as:

$$w_i \;\leq\; \frac{Wr_g(gH_{C-T}-g-1)}{(g+1)T(H_{C-T}-1) - (gH_{C-T}-g-1)W} = W - W\frac{r_g}{r_g\frac{n^2}{n-1} - W} \tag{4.9}$$

However, since $W \leq n \cdot r_g$, we have

$$W - W\frac{r_g}{r_g\frac{n^2}{n-1}-W} \;\geq\; W - W\frac{r_g}{r_g\frac{n^2}{n-1}-nr_g} = W - W\frac{n-1}{n^2-n(n-1)} = W - W\frac{n-1}{n} = \frac{W}{n}$$

Therefore, since $w_i \leq \frac{W}{n}$, inequality 4.9 holds and r_g is a better response for player i.

We now consider the case where $W > nr_g$. In this case, we have two subcases. (Note that in this case there is always some player k, such that $w_k \geq \frac{W}{n}$ and $w_k > d_g$.)

Case 1: There is a player i, such that $w_i \geq W/n$, $w_i > r_g$, and $W_{-i} + r_g \geq T$. In this case, Lemma 4.4.9 implies that r_g is a better response if and only if

$$(((g+1)T(H_{C-T}-1) - (gH_{C-T}-g-1)W)W_{-i} - Wr_g(gH_{C-T}-g-1)) \leq 0. \tag{4.10}$$

If again $(g+1)T(H_{C-T}-1) > (gH_{C-T}-g-1)W$, then the claim holds similarly with the case that $W \leq nd_g$. Otherwise, $(g+1)T(H_{C-T}-1) \leq (gH_{C-T}-g-1)W$, or

$$\frac{H_{C-T}W}{W + T(H_{C-T}-1)} \geq \frac{g+1}{g},$$

which implies

$$\frac{g+1}{g} \leq \frac{H_{C-T}W(r_g + W_{-i})}{W_{-i}(W + T(H_{C-T}-1)) + r_gW},$$

which is equivalent to inequality 4.10.

Case 2: For all j, such that $w_j > W/n$ and $w_j > r_g$, $W_{-j} + r_g < T$. Consider one such j and note that $W_{-j} < T - r_g < nr_g$ (for $g < (n-1)\frac{H_{C-T}}{H_{C-T}-1} + 1$). $W_{-i} + d_g < T$, which implies that $W_{-i} < T - r_g < (n-1)r_g$, for $g < n-1$, and $w_i = W - W_{-i} > nr_g - W_{-i}$. We will show that playing $nr_g - W_{-i}$ is a better response for i. But then the new total window size W' will be equal to $nr_g > T$ and thus, (according to the case that $W \leq nr_g$) there is another player j, with $w_j \leq \frac{W'}{n} and w_j < r_g$, that has r_g as a better response.

To see that $nr_g - W_{-i}$ is a better response for i, note that the derivative

$$\begin{aligned}\frac{\partial \pi_i^{RED2}(w)}{\partial w_i} &= \frac{\partial}{\partial w_i}\left(w_i \frac{\tilde{k}+T}{W}(g+1) - g w_i\right) \\ &= \left(\frac{w_i}{W H_{C-T}}\left(\frac{W-T}{H_{C-T}} + T\right)\left(\frac{1}{W} - \frac{w_i}{W^2}\right)\right)(g+1) - g\end{aligned}$$

has a unique positive root at $x_0 = br_{RED2}(w_{-i})$, and is negative and decreasing for all window sizes $x > x_0$. Moreover, its value when player i plays $nr_g - W_{-i}$ is

$$\begin{aligned}&\left(\frac{nr_g - W_{-i}}{nr_g H_{C-T}} + \left(\frac{nr_g - T}{H_{C-T}} + T\right)\left(\frac{1}{nr_g} - \frac{nr_g - W_{-i}}{n^2 r_g^2}\right)\right)(g+1) - g \\ &= \left(\frac{1}{H_{C-T}} + \frac{T W_{-i}(H_{C-T}-1)}{n^2 r_g^2 H_{C-T}}\right)(g+1) - g \\ &< \left(\frac{1}{H_{C-T}} + \frac{T(n-1) r_g (H_{C-T}-1)}{n^2 r_g^2 H_{C-T}}\right)(g+1) - g \\ &= \left(\frac{1}{H_{C-T}} + \frac{(H_{C-T}-1)(g H_{C-T} - g - 1)}{(g+1) H_{C-T}(H_{C-T}-1)}\right)(g+1) - g = 0\end{aligned}$$

Therefore, since $w_i > nr_g - W_{-i}$, we easily obtain that $\pi_i^{RED2}(w_i, w_{-i}) < \pi_i^{RED2}(nr_g - W_{-i}, w_{-i})$.

□

Lemma 4.4.11. *For any $w \neq (r_g, \ldots, r_g)$, there is a finite sequence of better replies that lead to the profile $(r_g, \ldots, r_g)$.*

Proof. We note first that if $W < T$, then for any player i, playing $T - W_{-i}$ is a better response than w_i. Hence we will assume that $W \geq T$. Note that applying Lemma 4.4.10 to $w \neq (r_g, \ldots, r_g)$, we will obtain some w' such that still $W' \geq T$. Therefore simply invoking Lemma 4.4.10 at most n times, we can see that there is a path of at most $2n+1$ better response moves from any profile $w \neq (r_g, \ldots, r_g)$ to the profile $(r_g, \ldots, r_g)$. □

We are now ready to prove our main theorem of this section.

Theorem 4.4.12. *If $g \in R_g^{hi}$, then the only stochastically stable state under RED is the state where all players set their window sizes to r_g.*

Proof. First note that r_g is the unique better response to $W_{-i} = (n-1)r_g$. Therefore the profile $d = (r_g, \ldots, r_g)$ is a minimal cusber set. Moreover, by Lemma 4.4.11, there is a better response path from any $w \neq d$ to d. Therefore any other cusber set would have to contain d, which implies there is no other minimal cusber set. Hence, by Corollary 4.2.3, d is the only state that is stochastically stable. □

The above theorem implies that under RED the system will converge to the unique Nash Equilibrium. Given that $g \in R_g^{hi}$, the total congestion will be less than the corresponding one in Droptail. Still, however, the overflow is large: as n grows, since $\frac{(g+1)(H_{C-T}-1)}{gH-g-1} > \frac{g+1}{g}$, the total window size will be (roughly) at least $2T$. And, as g decreases to values outside of R_g^{hi}, the congestion at RED NE can sometimes be even greater than at the Droptail NE. In the middle-ranged values of g, when $g < C/((C-T)(H_{C-T}-1))$, yet still large enough that $(r_g, \ldots, r_g)$ is the only NE, it is the case that $r_g > d_g$. So the NE under RED can sometimes lead to worse congestion at the router than the NE under Droptail.

4.5 "FAIR" QUEUE POLICY

In this section we study the queuing policy proposed in [46]. The main idea (similar also to the Prince algorithm in [28]) is that in case of congestion, the most demanding flow is punished. Assuming that all players are fully informed of the other players' strategies, this policy was constructed so as to have a unique NE in which all players share the capacity equally. In a more realistic setting, where the rates at which other flows send packets are not globally known, the authors wish to reach a less lofty goal: if all flows but one have fixed rates, then the unrestricted flow cannot use up much more of the router queue capacity at the expense of the fixed-rate flows. We will show here that in fact, the fair equilibrium is also the only stochastically stable state. This implies that, even without fully informed

players, the algorithm in [46] can achieve the fair NE, even when all flows are allowed to be arbitrarily aggressive.

The window game adaptation of Protocol I in [46] works as follows. For any profile $(w_1, \cdots, w_n)$, if $W \leq C$ then for any flow i, all w_i packets will enter the queue, i.e. $\pi_i(w) = w_i$. On the other hand, if $W > C$ then let $i_0 = \arg\max_{i \in N}\{w_i\}$ (breaking ties arbitrarily). Flow i_0 will be the one to be punished for the overflow, and if $w_{i_0} < W - C$ then the rest of the packets will be dropped according to Droptail. In other words, $\pi_{i_0} = \max\{0, w_{i_0} - (W - C)\} - g \cdot \min\{w_{i_0}, W - C\}$, while for any $i \neq i_0$,

$$\pi_i(w) = \begin{cases} w_i & \text{if } w_{i_0} \geq W - C \\ w_i \frac{C}{W - w_{i_0}} - g w_i \left(1 - \frac{C}{W - w_{i_0}}\right) & \text{if } w_{i_0} < W - C. \end{cases} \tag{4.11}$$

The next theorem was stated in [28].

Theorem 4.5.1. *Assuming $g > 0$, there is a unique NE in which all players play C/n.*

Proof. Note first that the profile $(C/n, \ldots, C/n)$ is a NE. If a flow reduces its window size, then it gets less packets in the queue and a decrease in its payoff. If it increases its window size, then again it will get only C/n packets in the queue, but it will also have dropped packets, which decrease its payoff.

We will now show that no profile $w \neq (C/n, \ldots, C/n)$ can be a NE. If $W < C$, any flow can increase its window size by $C - W$ and increase its payoff. Moreover, no state in which $W > C$ is a NE. If i is the flow such that $w_i = \max_{j \in N} w_j$, and thus gets punished for the overflow, then this flow can clearly benefit by reducing its window size. (The increase in the payoff happens either because i can get more packets through by reducing its window size, if there is another flow now that gets punished for the overflow, or because i gets the same number of packets in the queue, but saves on the cost incurred by dropped packets.) Therefore $W = C$ in any NE. Consider then a player i, such that $w_i = \min_{j \in N} w_j$. If $w_i < C/n$, given that $W = C$, there must be some player i' with $w_{i'} > C/n$. Then, flow i can increase its payoff by increasing its window size to C/n, since player i' will still be playing a higher value and thus get punished for i's increase. Therefore, in any NE w, $W = C$ and $w_j \geq C/n$, for all $j \in N$, implying that $w = (C/n, \ldots, C/n)$. □

The following theorem establishes the fact that the unique NE is also the only stochastically stable state. We prove this by showing that the state corresponding to the profile $(C/n, ..., C/n)$ is the only minimal cusber set.

Theorem 4.5.2. *If $g > 0$, then the only stochastically stable state corresponds to the profile $(C/n, ..., C/n)$.*

Proof. Let $\hat{w} = (C/n, \cdots, C/n)$. We will show that the singleton set $\{\hat{w}\}$ is *the only* minimal cusber set. Then we can conclude using Corollary 4.2.3 that $\hat{w}$ is the only stochastically stable state. First note that $\{\hat{w}\}$ is a minimal cusber set: any player deviating from $\hat{w}$ will be strictly decreasing her payoff. (Assume that a player i moves to some value $x \neq C/n$. If $x < C/n$, then $\pi_i(x, \hat{w}_{-i}) = x < C/n = \pi_i(\hat{w})$. If $x > C/n$, then $x - C/n$ of her packets will be dropped and her payoff will decrease to $\pi_i(x, \hat{w}_{-i}) = C/n - g(x - C/n) < \pi_i(\hat{w})$, since $g > 0$.)

We proceed now to showing that for any profile $w \neq \hat{w}$, there is a finite better response path to $\hat{w}$. Assume first that $W > C$ and let $i_0 = \arg\max_{i \in N}\{w_i\}$. Then $\min\{w_{i_0}, W - C\}$ of i_0's packets get dropped. In that case it is at least as good for i_0 to play $\max\{0, w_{i_0} - (W - C)\}$, since the same amount of i_0's packets will enter the queue as before, but without any being dropped. We will call this a *move of type A*.

Assume now that $W = C$, but $w \neq (C/n, \cdots, C/n)$. Let j be the player with the maximum window size in w, i.e., $j = \arg_{i \in N} \max w_i$. The fact that $W = C$ and $w \neq \hat{w}$, imply that $w_j > C/n$. Moreover there must be some player $k \neq j$, with $w_k < C/n$. Playing C/n is a better response to k, since $C/n < w_j$, meaning that j will be the one to be punished for the overflow. (The new total window size cannot exceed the capacity by more than C/n, implying only packets from flow j will be dropped.) Therefore, k gets more packets in the queue by changing w_k to C/n, and still none dropped. We will call this a *move of type B*.

Now the better response path to $\hat{w}$ is constructed as follows: From any w, if $W < C$, then any player can improve her payoff by increasing her window size by $C - W$. We then arrive at a state w' where $W' = C$. If $W > C$, after fewer than n moves of type A, a strategy profile w' is reached where $W' = C$.

For any w' such that $W' = C$, if $w' \neq \hat{w}$, then a move of type B occurs in which a player

that in w' played something less than C/n moves to C/n. This is immediately followed by a move of type A in which a player that in w' was playing something greater than C/n reduces her window size. If w'' corresponds to the new profile reached, then again $W'' = C$. This alternation between moves of type A and moves of type B continues, until $\hat{w}$ is reached. Note that once a player moves to C/n then she does not change her window size anymore, meaning that the total number of steps needed until $\hat{w}$ is reached is finite. □

We note that the condition $g > 0$ in the above theorem is necessary in order for the cusber set to contain only the profile $(C/n, \ldots, C/n)$. If $g = 0$, then a flow can deviate from the profile $(C/n, \ldots, C/n)$ by increasing its window size while still obtaining exactly the same payoff. We also note that unlike the results of Sections 4.3 and 4.4, the result in this section holds even if each flow has a different value for g, a value that can be arbitrarily small.

4.6 SUMMARY

In this chapter, we studied the long standing problem of congestion control at bottleneck routers on the internet. Many policies have been proposed for effective ways to drop packets from the queues of these routers so that network endpoints will be inclined to share router capacity fairly and minimize the overflow of packets trying to enter the queues. We studied just how effective some of these queuing policies are when each network endpoint is a self-interested player with no information about the other players' actions or preferences. By employing the adaptive learning model of evolutionary game theory, we examined policies such as Droptail, RED, and the greedy-flow-punishing policy proposed by Gao et al. [46] and found the stochastically stable states: the states of the system that will be reached in the long run.

We found that while Droptail and RED have stochastically stable states with high congestion at the bottleneck router, the Gao et al. policy leads to fair and efficient use of the bottleneck router capacity. Specifically, we've established that under Droptail queuing, the unique stochastically stable state (and unique NE) is the profile where all players send a

window size of $d_g = C(g+1)(n-1)/(gn^2)$. This means that if $g = (n-1)/(n+1)$, players will each be sending at least $2C/n$, twice as many total packets as capacity allows.

Under RED, when g is reasonably large enough (for $g \in R_g^{hi}$), the unique stochastically stable state (and unique NE) is the profile where all players send a window size of r_g, which is greater than $T(g+1)(n-1)/(gn^2)$. (Recall that $T < C$ is the threshold value at which RED begins preemptively dropping packets. It is a free parameter of the RED protocol.) This means, analogous to the above discussion about Droptail, if $g = (n-1)/(n+1)$ (which is close to 1 as n grows large), players will be sending at least $2T/n$. This would imply that when g is roughly 1, if deployers of RED routers set T to any more than $C/2$, endpoints will each send more than their fair share of C/n packets. And in general, for any $g \in R_g^{hi}$, if deployers set T to any more than roughly $Cg/(g+1)$, endpoints will each send more than C/n packets.

In contrast with RED then, a great benefit to the more discriminating Gao et al protocol is that it can be safely deployed without knowledge of the specific value of g: the endpoints each send C/n as long as g is positive. In addition, our results for the Gao et al protocol hold even when each player has a different, personal g value. Intuitively, this means the results apply to settings where the endpoints can be of all different types: well-behaved TCP flows, more aggressive TCP flows, UDP flows, etc.

A future possible direction of study would be to consider other variants on the utility function that are motivated by real-world internet traffic flows. One possible utility function to consider may be one that is tiered. When sending a video traffic flow, for example, there are certain packets that are far more valuable than others, so players may not profit uniformly over all packets, and likewise may not incur a uniform cost for all lost packets. This type of function is especially relevant since internet traffic is increasingly dominated by multi-media traffic. Another possibility is to consider concave utility functions, where a player's profit per successful packet diminishes as the number of successful packets nears the player's requested window size.

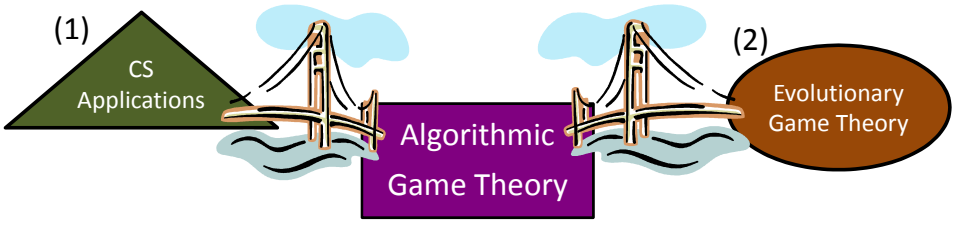
Another possible direction is that one might consider defining a variable g_i for each player i so that g, the penalty for each lost packet, need not be a global value. While perhaps more realistic (certain players/endpoints suffer a lot from each dropped packet, while others

may not mind a dropped packet very much at all), this generalization will make the game considerably more complicated to analyze. (While our results for the Gao et al protocol hold even when players have different, personalized g values, the results for Droptail and RED assume all players have the same g value.)

Finally, we find it curious that, for all three protocols, the unique Nash equilibrium was also the only stochastically stable state. We conjecture that our work thus far has been to prove special cases of a potentially more general theorem about the stochastically stable state of any protocol satisfying certain characteristics. In our future work, we plan to further explore this conjecture.

5.0 CONCLUSIONS

We sought to accomplish two goals in this work, the first to use game theoretic models for problems in applied computer science such as load balancing, data streams and internet traffic congestion, and the second to introduce the utility of adaptive learning from evolutionary game theory as an analytical and evaluative tool.



Towards both goals, in Chapter 2 we proposed the evolutionary game theory solution concept of stochastic stability as a tool for quantifying the relative stability of equilibria. We showed that in the load balancing game on unrelated machines, for which the price of Nash anarchy is unbounded, the "bad" Nash equilibria are not stochastically stable, and so the price of stochastic anarchy is bounded. We conjecture that the upper bound given in this chapter is not tight and the cost of stochastic stability for load balancing is $O(m)$. If this conjecture is correct, it implies that the fragility of the "bad" equilibria in this game is attributable to their instability, not only in the face of player coordination, but also to minor uncoordinated perturbations in play. We expect that the techniques used in this chapter will also be useful in understanding the relative stability of Nash equilibria in other games for which the worst equilibria are brittle. This promise is evidenced by the fact that the worst Nash in the worst-case instances in many games (for example, the Roughgarden and Tardos [74] lower bound showing an unbounded price of anarchy for routing unsplittable flow) are

not stochastically stable.

Towards our second goal, in Chapter 3 we applied techniques and principles from algorithmic game theory to a data streams query admission control problem. We introduced a new auction problem that, when posed abstractly, can be applied generally to naturally arising combinatorial settings outside of DSMSs. We introduced the notion of *sybil immunity* for auction mechanisms and proposed greedy and randomized auction mechanisms for this problem which are all *strategyproof*. We show the greedy approaches cannot give provable profit guarantees, and we also show that the randomized approach, for which we do give a provable profit guarantee, is not sybil immune. Our theoretical results pose the natural next question: can one find a mechanism for this problem that is strategyproof, sybil immune, *and* has a profit guarantee?

Our experimental results show that, generally speaking, CAT and CAF are the best mechanisms to use for profit maximization. However, if you have a high degree of operator sharing, and your system capacity is close to the total demand of the queries requesting service, then Two-price performs better for profit maximization. As expected, the greedy mechanisms (CAF, CAF+, CAT, and CAT+) provide better admission rate and payoff than Two-price. CAF+ and CAT+ are best for total user payoff, while CAF and CAF+ have the highest query admission rate as the degree of sharing increases. CAT, the one mechanism which is sybil immune, seems to offer the best overall tradeoff with respect to profit.

The data streams setting also lends itself to natural and interesting variants on our model. For example, we might consider a more general model where each query not only has a private valuation and a set of operators it wants executed, but it also has a specific interval of time during which it wants to be executed. The processing costs of each operator now represent costs per time unit, and the problem takes on a scheduling nature. Each time we choose to service a query we are committed to servicing it for the entire time interval specified by the query. Another possible variant is one where queries arrive over time, so the challenge becomes one of *online* mechanism design. And finally, we might consider the issue of energy consumption of the DSMS center. The profit of the DSMS center may not simply be the sum of user payments, but it may also incur a cost based on how much of the system is being utilized.

And finally, in Chapter 4, again towards both of our goals, we used evolutionary game theory to study the long standing problem of congestion control at bottleneck routers on the internet. Many policies have been proposed for effective ways to drop packets from the queues of these routers so that network endpoints will be inclined to share router capacity fairly and minimize the overflow of packets trying to enter the queues. We studied just how effective some of these queuing policies are when each network endpoint is a self-interested player with no information about the other players' actions or preferences. By employing the adaptive learning model of evolutionary game theory, we examined policies such as Droptail, RED, and the greedy-flow-punishing policy proposed by Gao et al. [46] and found the stochastically stable states: the states of the system that will be reached in the long run. We found that while Droptail and RED have stochastically stable states with high congestion at the bottleneck router, the Gao et al. policy leads to fair and efficient use of the bottleneck router capacity.

A future possible direction of study would be to consider other variants on the utility function that are motivated by real-world internet traffic flows. For example, when sending a video traffic flow, there are certain packets that are far more valuable than others, so players may not profit uniformly over all packets, and likewise may not incur a uniform cost for all lost packets. Hence, one possible utility function to consider may be one that is somehow tiered. This type of function is especially relevant since internet traffic is increasingly dominated by multi-media traffic. Another possibility is to consider concave utility functions, where a player's profit per successful packet diminishes as the number of successful packets nears the player's requested window size.

Finally, for all three protocols, the unique Nash equilibrium was also the only stochastically stable state. It is possible that our work thus far has been only to prove special cases of a potentially more general theorem about the stochastically stable state of any router queuing protocol satisfying certain characteristics. Further exploration of this possibility is needed.

With these three works as a whole, we have made inroads toward building bridges between algorithmic game theory, evolutionary game theory's adaptive learning, and applied computer science.

BIBLIOGRAPHY

[1] Daniel J. Abadi, Don Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. Aurora: a new model and architecture for data stream management. *The VLDB Journal*, 12(2):120–139, 2003.

[2] Gagan Aggarwal and Jason D. Hartline. Knapsack auctions. In *SODA*, 2006.

[3] Aditya Akella, Srinivasan Seshan, Richard M. Karp, Scott Shenker, and Christos H. Papadimitriou. Selfish behavior and stability of the internet: a game-theoretic analysis of tcp. In *SIGCOMM*, 2002.

[4] Susanne Albers. On the value of coordination in network design. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 294–303, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[5] Susanne Albers, Stefan Eilts, Eyal Even-Dar, Yishay Mansour, and Liam Roditty. On nash equilibria for a network creation game. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 89–98, New York, NY, USA, 2006. ACM.

[6] Nir Andelman, Michal Feldman, and Yishay Mansour. Strong price of anarchy. In *SODA '07*.

[7] Elliot Anshelevich, Anirban Dasgupta, Jon Kleinberg, va Tardos, Tom Wexler, and Tim Roughgarden. The price of stability for network design with fair cost allocation. In *In FOCS*, pages 295–304, 2004.

[8] Elliot Anshelevich, Anirban Dasgupta, Eva Tardos, and Tom Wexler. Near-optimal network design with selfish agents. In *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 511–520, New York, NY, USA, 2003. ACM.

[9] Baruch Awerbuch, Yossi Azar, and Amir Epstein. The price of routing unsplittable flow. In *STOC '05*.

[10] Baruch Awerbuch, Yossi Azar, Yossi Richter, and Dekel Tsur. Tradeoffs in worst-case equilibria. *Theor. Comput. Sci.*, 361(2):200–209, 2006.

[11] Brian Babcock, Mayur Datar, and Rajeev Motwani. Load shedding for aggregation queries over data streams. In *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*, page 350, Washington, DC, USA, 2004. IEEE Computer Society.

[12] Stephen Baker. Google and the wisdom of clouds. *Business Week*, December 2007.

[13] Avrim Blum, Eyal Even-Dar, and Katrina Ligett. Routing without regret: On convergence to Nash equilibria of regret-minimizing algorithms in routing games. In *PODC '06*.

[14] Avrim Blum, MohammadTaghi Hajiaghayi, Katrina Ligett, and Aaron Roth. Regret minimization and the price of total anarchy. In *STOC '08*.

[15] Lawrence E. Blume. The statistical mechanics of best-response strategy revision. *Games and Economic Behavior*, 11(2):111–145, November 1995.

[16] Liad Blumrosen and Noam Nisan. Combinatorial auctions. In Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, 2007.

[17] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Larry Peterson Partridge, K. K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. RFC2309: Recommendations on queue management and congestion avoidance in the internet. *Internet RFCs*, 1998.

[18] Ho-Lin Chen and Tim Roughgarden. Network design with weighted players. In *SPAA '06: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 29–38, New York, NY, USA, 2006. ACM.

[19] Xi Chen and Xiaotie Deng. Settling the complexity of 2-player Nash-equilibrium. In *FOCS '06*.

[20] George Christodoulou and Elias Koutsoupias. The price of anarchy of finite congestion games. In *STOC '05*.

[21] Christine Chung, Katrina Ligett, Kirk Pruhs, and Aaron Roth. The price of stochastic anarchy. In *SAGT*, 2008.

[22] Roberto Cominetti, Jos R. Correa, and Nicols E. Stier Moses. Network games with atomic players. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (1)*, volume 4051 of *Lecture Notes in Computer Science*, pages 525–536. Springer, 2006.

[23] Jacomo Corbo and David Parkes. The price of selfish behavior in bilateral network formation. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 99–107, New York, NY, USA, 2005. ACM.

[24] Artur Czumaj and Berthold Vöcking. Tight bounds for worst-case equilibria. In *SODA '02*.

[25] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 1–12, 1989.

[26] C. Douligeris and R. Mazumdar. On Pareto optimal flow control in an integrated environment. In *Allerton Conference on Communication, Control and Computing*, 1987.

[27] C. Douligeris and R. Mazumdar. A game theoretic approach to flow control in an integrated environment. *Journal of the Franklin Institute*, 329(3):383–402, 1992.

[28] P.S. Efraimidis and L. Tsavlidis. Window-games between tcp flows. In *The first international symposium on algorithmic game theory*, 2008.

[29] Glenn Ellison. Basins of attraction, long-run stochastic stability, and the speed of step-by-step evolution. *Review of Economic Studies*, 67(1):17–45, January 2000.

[30] Amir Epstein, Michal Feldman, and Yishay Mansour. Strong equilibrium in cost sharing connection games. In *EC '07: Proceedings of the 8th ACM conference on Electronic commerce*, pages 84–92, New York, NY, USA, 2007. ACM.

[31] Eyal Even-Dar, Alexander Kesselman, and Yishay Mansour. Convergence time to Nash equilibria. In *ICALP '03*.

[32] Alex Fabrikant, Ankur Luthra, Elitza N. Maneva, Christos H. Papadimitriou, and Scott Shenker. On a network creation game. In *PODC*, pages 347–351, 2003.

[33] Alex Fabrikant and Christos Papadimitriou. The complexity of game dynamics: Bgp oscillations, sink equilibria, and beyond. In *SODA '08*.

[34] Alex Fabrikant, Christos Papadimitriou, and Kunal Talwar. The complexity of pure nash equilibria. In *STOC '04*.

[35] Uriel Feige, David Peleg, and Guy Kortsarz. The dense -subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[36] A. Fiat, A.V. Goldberg, J.D. Hartline, and A.R. Karlin. Competitive generalized auctions. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of Computing*, pages 72–81. ACM New York, NY, USA, 2002.

[37] Amos Fiat, Andrew V. Goldberg, Jason D. Hartline, and Anna R. Karlin. Competitive generalized auctions. In *STOC*, 2002.

[38] Amos Fiat, Haim Kaplan, Meital Levy, and Svetlana Olonetsky. Strong price of anarchy for machine load balancing. In *ICALP '07*.

[39] Amos Fiat, Haim Kaplan, Meital Levy, Svetlana Olonetsky, and Ronen Shabo. On the price of stability for designing undirected networks with fair cost allocations. In *ICALP (1)*, pages 608–618, 2006.

[40] Simon Fischer, Harald Räcke, and Berthold Vöcking. Fast convergence to wardrop equilibria by adaptive sampling methods. In *STOC '06*.

[41] Simon Fischer and Berthold Vöcking. On the evolution of selfish routing. In *ESA '04*.

[42] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1(4):397–413, 1993.

[43] D. Foster and P. Young. Stochastic evolutionary game dynamics. *Theoret. Population Biol.*, 38:229–232, 1990.

[44] Dimitris Fotakis, Spyros Kontogiannis, and Paul Spirakis. Selfish unsplittable flows. *Theor. Comput. Sci.*, 348(2):226–239, 2005.

[45] Eric Friedman, Paul Resnick, and Rahul Sami. Manipulation-resistant reputation systems. In *Algorithmic Game Theory*. 2007.

[46] X. Gao, K. Jain, and L.J. Schulman. Fair and efficient router congestion control. In *SODA 2004*.

[47] Rahul Garg, Abhinav Kamra, and Varun Khurana. A game-theoretic approach towards congestion control in communication networks. *SIGCOMM Comput. Commun. Rev.*, 32(3):47–61, 2002.

[48] Michel Goemans, Vahab Mirrokni, and Adrian Vetta. Sink equilibria and convergence. In *FOCS '05*.

[49] Andrew V. Goldberg, Jason D. Hartline, and Andrew Wright. Competitive auctions and digital goods. In *SODA*, 2001.

[50] A.V. Goldberg and J.D. Hartline. Envy-free auctions for digital goods. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 29–35. ACM New York, NY, USA, 2003.

[51] A.V. Goldberg, J.D. Hartline, A.R. Karlin, M. Saks, and A. Wright. Competitive auctions. *Games and Economic Behavior*, 55(2):242–269, 2006.

[52] The STREAM Group. Stream: The stanford stream data manager. *IEEE Data Engineering Bulletin*, 2003.

[53] V. Jacobson. Congestion avoidance and control, ACM SIGCOMM, 1988.

[54] Jens Josephson and Alexander Matros. Stochastic imitation in finite games. *Games and Economic Behavior*, 49(2):244–259, November 2004.

[55] Michihiro Kandori, George J Mailath, and Rafael Rob. Learning, mutation, and long run equilibria in games. *Econometrica*, 61(1):29–56, January 1993.

[56] A. Kesselman. St. Leonardi, and V. Bonifaci. Game-theoretic Analysis of Internet Switching with Selfish Users. In *WINE 2005*.

[57] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413, Trier, Germany, 4–6 March 1999.

[58] Samuelson Larry. Stochastic stability in games with alternative best replies. *Journal of Economic Theory*, 64(1):35–65, October 1994.

[59] H.J. Lee and J.T. Lim. Performance Analysis of CHOKe with Multiple UDP Flows. In *SICE-ICASE, 2006. International Joint Conference*, pages 5200–5203, 2006.

[60] Daniel J. Lehmann, Liadan O'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. *J. ACM*, 49(5):577–602, 2002.

[61] Jian Li. An $O(\log n/\log\log n)$ upper bound on the price of stability for undirected shapley network design games. Manuscript (arXiv:0812.2567v1), 2008.

[62] Marios Mavronicolas and Paul G. Spirakis. The price of selfish routing. In *STOC '01*.

[63] Paul McDougall. Google, ibm join forces to dominate 'cloud computing'. *Information Week*, May 2009.

[64] D. S. Menasché, Daniel R. Figueiredo, and Edmundo de Souza e Silva. An evolutionary game-theoretic approach to congestion control. *Perform. Eval.*, 62(1-4):295–312, 2005.

[65] Ahuva Mu'alem and Noam Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *AAAI/IAAI*, pages 379–384, 2002.

[66] Noam Nisan. *Introduction to Mechanism Design*, chapter 9, pages 209–241.

[67] Noam Nisan and Amir Ronen. Algorithmic mechanism design. In *Games and Economic Behavior*, pages 129–140, 1999.

[68] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[69] Chris Olston, Jing Jiang, and Jennifer Widom. Adaptive filters for continuous queries over distributed data streams. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 563–574, New York, NY, USA, 2003. ACM.

[70] Rong Pan, Balaji Prabhakar, and Konstantinos Psounis. Choke, a stateless active queue management scheme for approximating fair bandwidth allocation. In *INFOCOM*, pages 942–951, 2000.

[71] Anita Ramasastry. Web sites change prices based on customers' habits, June 2005. http://www.cnn.com/2005/LAW/06/24/ramasastry.website.prices/.

[72] Spencer Reiss. Cloud computing. available at amazon.com today. *Wired*, April 2008.

[73] Arthur J. Robson and Fernando Vega-Redondo. Efficient equilibrium selection in evolutionary games with random matching. *Journal of Economic Theory*, 70(1):65–92, July 1996.

[74] Tim Roughgarden and Éva Tardos. How bad is selfish routing? *J. ACM*, 49(2):236–259, 2002. Also appeared in FOCS 2000.

[75] Mohamed A. Sharaf, Panos K. Chrysanthis, Alexandros Labrinidis, and Kirk Pruhs. Algorithms and metrics for processing multiple heterogeneous continuous queries. *ACM Trans. Database Syst.*, 33(1):1–44, 2008.

[76] SJ Shenker. Making greed work in networks: a game-theoretic analysis of switchservice disciplines. *IEEE/ACM Transactions on Networking*, 3(6):819–831, 1995.

[77] Yoav Shoham. Computer science and game theory. *Communications of the ACM*, 51(8), August 2008.

[78] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks. In *ACM SIGCOMM'98*.

[79] David Streitfeld. On the web, price tags blur: What you pay could depend on who you are, September 2000. http://www.washingtonpost.com/ac2/wp-dyn/A15159-2000Sep25.

[80] Siddarth Suri. Computational evolutionary game theory. In Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, 2007.

[81] Subhash Suri, Csaba D. Tóth, and Yunhong Zhou. Selfish load balancing and atomic congestion games. In *SPAA '04*.

[82] A. Tang, J. Wang, and S.H. Low. Understanding CHOKe: throughput and spatial characteristics. *IEEE/ACM TRANSACTIONS ON NETWORKING*, 12(4), 2004.

[83] Nesime Tatbul, Uğur Çetintemel, Stan Zdonik, Mitch Cherniack, and Michael Stonebraker. Load shedding in a data stream manager. In *VLDB Conf*, 2003.

[84] Yi-Cheng Tu, Song Liu, Sunil Prabhakar, and Bin Yao. Load shedding in stream databases: a control-based approach. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 787–798. VLDB Endowment, 2006.

[85] Adrian Vetta. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *In Proc. of IEEE FOCS*, pages 416–425, 2002.

[86] Troy Wolverton. Amazon backs away from test prices, September 2000. http://news.cnet.com/2100-1017-245631.html.

[87] H Peyton Young. The evolution of conventions. *Econometrica*, 61(1):57–84, January 1993.

[88] H Peyton Young. *Individual Strategy and Social Structure*. Princeton University Press, Princeton, NJ, 1998.