Fundamental Limit of Network Flow Attacks

by

Xinzhe Fu

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Signature redacted

Author ....
Department of Aeronautics and Astronautics
May 23, 2019

Signature redacted

Certified by...................... ....
Eytan Modiano
Professor of Aeronautics and Astronautics
Thesis Supervisor

Signature redacted

Accepted by.................... ....
Sertac Karaman
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Fundamental Limit of Network Flow Attacks

by

Xinzhe Fu

Submitted to the Department of Aeronautics and Astronautics
on May 23, 2019, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

A network flow-based attack refers to a cyber-attack where the adversary seeks to block user traffic from transmission by sending adversarial traffic that reduces the available user capacity. In this thesis, we explore the fundamental limits of network flow attacks by investigating its feasibility region defined by the minimum resource required for a successful attack and designing optimal attacking strategies that achieve the feasibility region.

First, we consider the case where the target network uses fixed-path routing and the adversary injects traffic into the network, encroaching the capacity of the network links and thus reducing the capacity available to network users on the fixed paths. We propose a new network interdiction paradigm that captures this phenomenon by modeling the network as a capacitated graph with the user throughput given by the max-flow value on the fixed user paths. The adversary injects interdicting flows that reduces the capacity of the links (and hence the user throughput), and seeks to maximize the throughput reduction caused by the adversarial injection under a given flow budget. We show the NP-hardness of the problem of maximizing throughput reduction, and propose an efficient approximation algorithm that yields near optimal interdicting flows within a logarithmic factor by harnessing the submodularity of the problem. We further extend the algorithm to an approximation framework that can deal with the situation where the adversary does not have deterministic knowledge of the set of user paths but aims to maximize the worst case throughput reduction given that the set of user paths lies in certain collection of paths.

Next, we turn to the scenario where the target network employs dynamic routing mechanisms such as Join-the-Shortest-Queue (JSQ) or Max-Weight. We start from single-hop server farm under JSQ routing, where the adversary attacks by injecting adversarial traffic to servers with the objective of blocking user traffic, i.e., causing user traffic to experience unbounded delay. We first characterize the feasibility region of the attack by presenting a necessary and sufficient condition on the rate of adversarial traffic rate for the attack to be successful. We then propose an adversarial injection policy that is, (i) *optimal:* it achieves a successful attack whenever the adversarial traffic rate is inside the feasibility region and (ii) *oblivious:* it does not rely on any knowledge of the network statistics. We further evaluate the performance of the injection policy. Finally, we extend our results to multi-hop network employing Max-Weight routing.

Thesis Supervisor: Eytan Modiano
Title: Professor of Aeronautics and Astronautics

Acknowledgments

I greatly acknowledge the invaluable guidance and support of my thesis supervisor, Professor Eytan Modiano, whose patience and kindness have helped me navigate the ups and downs of my master's studies.

I am blessed to have made amazing friends at MIT, who have helped me a lot, both in study and in life. I really enjoyed my discussion on research and coursework with Vishrant Tripathi, Bai Liu, Tianyi Peng, Zehao Yu, Igor Kadota, Ruihao Zhu, Xinyu Wu, Ertem Nusret Tas, Georgia Dimaki, Rajat Talak, Dr. Jianan Zhang, Dr. Qingkai Liang, Dr. Thomas Stahlbuhk, Dr. Anurag Rai, Dr. Dan Wu, Prof. Hyang-Won Lee and Prof. Abhishek Sinha. As sports, especially tennis and soccer, has been an integral part of my life and a major source of happiness, I am grateful to my previous and current tennis coaches, Nick White and Charlie Maher, my tennis pals, soccer teammates and others who have made my entertaining sporting experience at MIT. I very much appreciate friends that I have known since high school or college, Shengzhong Liu, Shucen Liu, Mengdie Zhou, Zhiying Xu, Prof. Luoyi Fu and Prof. Xinbing Wang for letting me share my feelings and thoughts. I also want to thank my roommate, Yuanming Hu, for our pleasant two years sharing a home.

I want to extend my deepest gratitude to my family. In particular, I am very grateful to my uncle Sheldon He and his family (Wei Su He, Harrison He, Felicia He) for hosting me during Thanksgiving and Christmas holidays and giving me advice on my life in the US. Finally, I am forever indebted to my parents, Xiaoxin He and Yonghui Fu, who have always been there for me no matter what we are going through.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Network flow attacks refer to a class of cyber-attacks where the adversary seeks to make some network resources unavailable to its intended users by sending adversarial traffic flow that competes for the resources. On one hand, attacks of this category, such as DoS attack [1], stealth DoS attack [2], RoQ attack [3] and Pulsing attack [4] often result in downtime of web service, cloud computing, DNS service, etc., causing huge financial loss to institutions [5]. On the other hand, network flow-based attack is difficult to detect due to the similarity of adversarial traffic and normal user traffic. Moreover, the increasing availability of botnet [6] as sources of adversarial traffic exacerbates the issue. The above two aspects make network flow-based attack one of the most serious security threat to the Internet. Due to the significance and prevalence of network flow attacks, there have been a flurry of works focusing on detection and mitigation of such attacks [1,7,8]. However, we still lack theoretical understanding of the limits of such attacks, i.e., **How much resources does the adversary need for a mounting successful network flow attack and what is the optimal attack strategy?**

Understanding the above questions is of great importance to the design and protection of network systems. It provides us with valuable insights on the robustness of the network system as the resource requirement for the adversary actually defines the safety margin of the system. Such safety margin often plays an important role in evaluating the vulnerabilities of network infrastructure [9, 10]. Furthermore, the structure of the optimal attack strategy sheds light on the design of practical detection and mitigation methods, providing guidelines to the design of security-enhanced systems [11].

In this thesis, we explore the fundamental limits of the network flow attacks. Taking a network flow and queueing perspective, we model the attack using a capacitated network with user traffic and adversarial traffic generated at certain rates from certain sources, respectively. The user traffic is routed using some pre-defined policy of the network, while the adversarial traffic is injected according to some policy dictated by the adversary. The goal of the adversary is to reduce the throughput available to user traffic by injecting adversarial traffic in an intelligent way. We interpret the link capacities and user traffic rates as network parameters, and the adversarial traffic rates as the budget of the adversary. Based on this, the two problem we raised, which will be the central theme of the thesis can be translated into: given the network parameters, how much budget does it take to achieve a successful attack, and what is the optimal adversarial injection policy that guarantees the success or maximize the damage of an attack?

1.2 Thesis Contribution

We answer the questions through the following main results, which can be divided into two parts based on the routing policy that the network uses.

- In Chapter 2 we study the case where the network employs fixed-path routing, which captures routing protocols such as shortest-path routing (OSPF, BGP) and multi-path routing (ECMP). Under fixed-path routing, the problem falls into the category of network interdiction which models the scenario where a budget-constrained interdictor tries to reduce the throughput available for *users* of a capacitated network [12, 13]. Since the traditional interdiction paradigm falls short of capturing the flow-based attack as it only considers network node/link removal, we propose a new network interdiction paradigm where the interdiction is performed by injecting adversarial traffic flow. Under the proposed paradigm, we study the computational complexity of the problem and devise a near-optimal interdiction strategy that approximately maximizes the throughput reduction.

- In Chapter 3 we investigate the scenario where the network employs dynamic routing strategy such as Join the Shortest Queue and Max-Weight [14–16], which naturally mirror systems with load-balancing mechanisms such as server farms. We define the feasibility region of the network flow attack under this setting and design an optimal attack strategy that achieves the full feasibility region. A notable feature of the optimal strategy is that it does not rely on knowledge of network statistics.

1.3 Literature Review

We present a brief literature review on three relevant topics, adversarial network optimization and network attack-defense framework.

1.3.1 Network Interdiction

Network interdiction, originally proposed in [12, 17] models the scenarios where a budget-constrained *interdictor* tries to limit the throughput available for *users* of a capacitated network by removing network edges. The throughput is given by the optimal value of a single-commodity max-flow problem and the goal of the interdictor is to compute an interdiction strategy that specifies which edges to remove in order to minimize the throughput, or maximize the throughput reduction, subject to the budget constraint. Since the problem is NP-hard even when the network has special topologies, previous works focus on designing approximation algorithms [17–19] or formulating integer programs and solving them using traditional optimization techniques (e.g. branch and bound) [12,20]. Subsequent generalizations include extensions to the case where the throughput is given by multi-commodity max-flow problem [21] and allowing the interdictor to use mixed strategy that takes advantage of randomization [22, 23]. We refer the readers to [24] for a comprehensive survey. As traditional network interdiction only considers link removal, it cannot capture the scenario of network flow attacks where the method of attack is injecting traffic flow rather than removing network links.

1.3.2 Adversarial Network Optimization

Adversarial network optimization studies the problem of optimizing network performance in adversarial environments. This area of study started from adversarial queueing theory [25] that investigates the stability of various service disciplines under adversarial-generated traffic. Previous works [25–27] established the universal stability of several service disciplines such as Newest-in-System and Furthest-to-Go, the instability of service disciplines such as First-Come-First-Serve and the universal stability of certain network topologies. Later, Lim et al. [28] extended the framework to multi-hop network routing and showed the throughput optimality of Max-Weight policy under adversarial traffic. Recently, Liang and Modiano extended the theory to optimizing utility/queue lengths in networks with adversarial traffic and network states [29, 30]. They analyzed the performance of Max-Weight policy and Tracking policy in terms of regret, establishing the optimality of the two policies under certain adversarial network models. Adversarial network optimization considers the case where there is only adversarial traffic whereas we consider a mix of both adversarial and user traffic. Another major difference is that in adversarial network optimization, although the traffic is adversarial, it is still assumed to be stabilizable, i.e., lies in the capacity region of the network. In contrast, in our setting of network DoS attack, the case of interest is often the overload regime, where the combined traffic of user and adversary lies outside of the capacity region.

1.3.3 Network Attack-Defense Framework

Network attack-defense framework investigates the optimal (attack or defense) strategy in a network with two parties of conflicting interests. Most works in this line adopt a game-theoretic perspective [31–33]. Wang and Shroff [31] focused on equilibrium analysis of network security game. Niyato and Hossain [33] studied the dynamics of network selection through an evolutionary game. For a comprehensive review on this topic, we refer the reader to the survey by Manshaei et al. [32] The modeling approach of game theory does not possess the fine-granularity to capture important aspects in flow-based attack such as network routing and network dynamics. An important exception that does not use game-theoretic modeling is the paper Paschos and Tassiulas [2]. They studied the sustainability of service provisioning system, which is similar to a special case of our problem on single-hop networks. Apart from network settings, another important distinction is that they did not investigate the problem of designing optimal attack strategies.

Chapter 2

Network Interdiction Using Adversarial Traffic Flow

In this chapter, we study network flow attacks where the network employs fixed-path routing. As the problem falls into the broad category of network interdiction, we propose a new network interdiction paradigm where the interdiction is performed through injecting adversarial traffic flow to the network in an intelligent way, encroaching the capacity of network links, thereby reducing the throughput of network users. Our models captures is the stealth denial of service (DoS) attack in communication networks (including wireless ad hoc networks [34], software defined networks [35] and cloud services [36]), where interdictor (attacker) injects low-rate data into the network that consumes network resources and compromises the capacity available to the users. Under this paradigm, we model the network as a capacitated directed graph with n nodes, where the network users are sending flow on a set P of user paths and the interdictor aims to reduce the throughput of the users through sending adversarial flow from its source s to its destination t. Mirroring the situations in [2, 34, 35], we assume the interdictor to be low-rate and undetectable, which will be formally defined later. The interdiction strategy is defined as a probability distribution over the set of s-t flows with value less than the given budget, which resembles the mixed strategy in the game theory literature [22]. The throughput reduction achieved is equal to the difference between the network throughput before the interdiction, which is defined as the sum of initial flow values on paths in P and the network throughput after the interdiction, which is determined by the optimal value of a path-based max-flow problem on the residual network.

Under the proposed interdiction paradigm, we study two problems that differ in the availability of the knowledge on the operation of network users captured by the set of user paths P. The first, *deterministic flow interdiction*, assumes that the interdictor has perfect knowledge of P and seeks an interdiction strategy that maximizes the (expected) throughput reduction with respect to P. We show that there does not exist any polynomial time algorithm that approximates the problem on general networks within an $O(n^{1-\delta})$ factor for any $\delta > 0$ unless P = NP, and the problem is NP-hard even when the network is acyclic. Thus, we focus on designing efficient algorithms with good performance guarantees on acyclic networks. Specifically, utilizing the submodularity of the problem, we propose a recursive algorithm that is capable of achieving $O(\log n)$-approximation. The second problem, *robust flow interdiction*, deals with the situation where definitive knowledge of P is not available. In particular, we assume that the set of user paths lies in some uncertainty set $\mathcal{U}$ that contains all possible candidates for P. The goal of the interdictor is to compute an interdiction strategy that maximizes the throughput reduction for the worst case in $\mathcal{U}$. As a generalization of its deterministic counterpart, robust flow interdiction inherits the computational complexity results and is more challenging to solve due to its inherent maximin objective. In this context, we design an approximation framework that integrates the algorithm for deterministic flow interdiction and yields a quasi-polynomial time procedure with a poly-logarithmic approximation guarantee. Finally, We evaluate the performance of the proposed algorithms through simulations. The simulation results suggest that our algorithms compute solutions that are at least 70% of the optimal and are efficiently implementable.

The rest of the chapter is organized as follows. We formally present our paradigm on acyclic networks in Section 2.1. In Sections 2.2 and 2.3, we introduce formal definitions, show the computational complexity and describe our proposed algorithms for the two flow interdiction problems, respectively. We evaluate the performance of our algorithms through simulations in Section 2.4. Section 2.5 is devoted to the extension of our paradigm and the interdiction problems to general networks. We conclude the chapter in Section 2.6.

2.1 Network Interdiction Paradigm

In this section, we first formalize our network interdiction paradigm, and then show two important structural properties of it. Note that currently, we focus on acyclic networks, and provide extensions to general networks in Section 2.5.

Consider a network represented as a directed acyclic graph $G(V, E)$ with vertex set V and edge set $E \subseteq V \times V$. Let $n = |V|$ be the number of nodes and $m = |E|$ be the number of edges. We assume G to be simple (with no multi-edges). Let C be an $|E|$-dimensional non-negative capacity vector with $C(e)$ indicating the capacity of edge e. We define $s, t \in V$ as the source and the destination of the interdictor, and assume without loss of generality that they are connected. An s-t flow is defined as an $|E|$-dimensional vector $\mathbf{f}$ that satisfies capacity constraints: $\forall e \in E, 0 \leq \mathbf{f}(e) \leq C(e)$ and flow conservation constraints: $\forall v \in V \backslash \{s, t\}, \sum_{(u,v)\in E} \mathbf{f}(u, v) = \sum_{(v,u)\in E} \mathbf{f}(v, u)$. We define $val(\mathbf{f}) = \sum_{(s,u)\in E} \mathbf{f}(s, u)$ to be the value of $\mathbf{f}$, i.e., the total flow out of the source.

The interdiction is performed by injecting flow from s to t. The interdictor has a flow budget γ that specifies the maximum value of flow that it can inject. In this paper, we are primarily concerned with low-rate interdictor, and thus assume that $\gamma \leq \min_{e\in E} C(e)$ and is bounded by some polynomial of n. Let $\mathcal{F}_{\leq\gamma}$ be the set of s-t flows $\mathbf{f}$ with $val(\mathbf{f}) \leq \gamma$. We allow the interdictor to use randomized flow injection, which is captured by the concept of interdiction strategy formally defined below.

Definition 1 (Interdiction Strategy). *An interdiction strategy w is a probability distribution $w : \mathcal{F}_{\leq\gamma} \mapsto [0, 1]$ such that $\sum_{\mathbf{f}\in\mathcal{F}_{\leq\gamma}} w(\mathbf{f}) = 1$*

The interdiction strategy bears resemblance to the mixed strategy in the game theory literature. It can be alternatively interpreted as injecting flows in a time sharing way. Furthermore, a deterministic flow injection $\mathbf{f}$ (similar to a pure strategy in game theory) can be represented as a strategy with $w(\mathbf{f}) = 1$.

Before the interdiction, the network users are sending flow on a set of user paths $P = \{p_1, p_2, \dots, p_k\}$ in the network. Each path is a subset of edges and we use $e \in p_i$ to represent that edge e is on path p_i. The user paths may not share the same source and destination, and are not necessarily disjoint. Initially, the values of the flows on the paths are $\lambda_1, \lambda_2 \dots, \lambda_k$ respectively, which satisfy capacity constraints: $\forall e \in E$, $\sum_{p_i \ni e} \lambda_i \leq C(e)$. The network throughput for the users before the interdiction is defined as $\sum_{i=1}^{k} \lambda_i$. Note that the involvement of the initial flows gives our paradigm the flexibility to capture the case where the users are not fully utilizing the paths before interdiction.

After the interdictor injects flow $\mathbf{f}$, the residual capacity of the edges becomes $\tilde{C}_\mathbf{f}$ such that $\tilde{C}_\mathbf{f}(e) = C(e) - \mathbf{f}(e)$ for all $e \in E$. The throughput of the users after interdiction is given by the optimal value of the following (path-based) max-flow problem:

$$\text{maximize } \sum_i \tilde{\lambda}_i \tag{2.1}$$

$$\textbf{s.t. } \sum_{p_i \ni e} \tilde{\lambda}_i \leq \tilde{C}_\mathbf{f}(e), \quad \forall e \in E \tag{2.2}$$

$$0 \leq \tilde{\lambda}_i \leq \lambda_i, \quad \forall i \tag{2.3}$$

where constraints (2.2) are the capacity constraints after the interdiction and constraints (2.3) specify that the users will not actively push more flows on the paths after the interdiction, which can be attributed to the undetectability of the interdictor or that the users have no more flow to send. Let $T(\mathbf{f}, P)$ be the optimal value of (2.1). We define the throughput reduction achieved by injecting flow $\mathbf{f}$ as the difference between the throughput of the network before and after interdiction, i.e., $\Lambda(\mathbf{f}, P) = \sum_i \lambda_i - T(\mathbf{f}, P)$. Naturally, under an interdiction strategy w, the expected throughput reduction achieved by the interdictor is defined as $\Lambda(w, P) = \sum_{\mathbf{f} \in \mathcal{F}_{\leq \gamma}} w(\mathbf{f}) \Lambda(\mathbf{f}, P)$.

2.1.1 Structural Properties of the Paradigm

The proposed network interdiction paradigm has two important structural properties that will play a key role in the problems we study in subsequent sections. The first property shows that if we want to maximize the throughput reduction, we can restrict our consideration to the set of s-t flows with value γ. Its proof follows straightforwardly from the monotonicity of throughput reduction with respect to the value of the interdicting flow and that $\gamma \leq \min_e C(e)$.

Observation 1. *For any s-t flow $\mathbf{f}$ with $val(\mathbf{f}) < \gamma$, there exists a flow $\mathbf{f}'$ such that $val(\mathbf{f}') = \gamma$ and $\Lambda(\mathbf{f}', P) \geq \Lambda(\mathbf{f}, P)$ for all possible P.*

We denote $\mathcal{F}_\gamma$ to be the set of flows with value γ. We further define *single-path flows* as the s-t flows that have positive values on edges of one s-t path. The second property establishes the optimality of interdiction strategies taking positive value on only single-path flows in the maximization of Λ.

Proposition 1. *For any interdiction strategy w, there exists an interdiction strategy w' that is a probability distribution on the set of single-path flows such that $\Lambda(w', P) \geq \Lambda(w, P)$ for all possible P.*

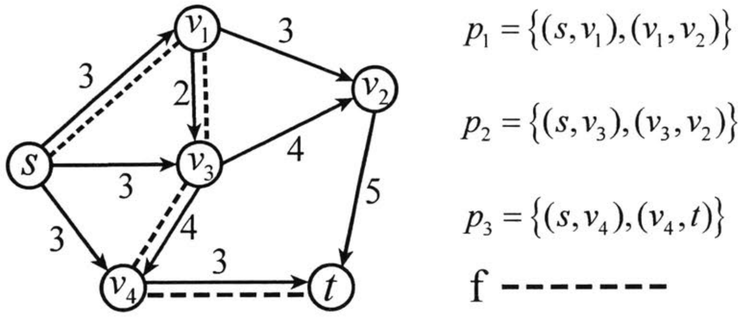
Proof. We prove the proposition through flow decomposition and linear programming duality. See Appendix 2.7.1 for details. $\square$

2.2 Deterministic Flow Interdiction

In this section, we study the deterministic flow interdiction problem. We first formally define the problem, then prove its computational complexity, and finally introduce our proposed approximation algorithm.

2.2.1 Problem Formulation

The deterministic flow interdiction deals with the case where the interdictor has full knowledge of P and seeks the interdiction strategy that causes the maximum expected throughput reduction.



$p_1 = \{(s, v_1), (v_1, v_2)\}$
$p_2 = \{(s, v_3), (v_3, v_2)\}$
$p_3 = \{(s, v_4), (v_4, t)\}$
f

Figure 2-1: An example network of the flow interdiction problems.

Definition 2 (Deterministic Flow Interdiction). *Given the set of user paths $P = \{p_1, \ldots, p_k\}$ with initial flow values $\{\lambda_1, \ldots, \lambda_k\}$, the deterministic flow interdiction problem seeks an interdiction strategy w that maximizes $\Lambda(w, P)$.*

Example: We give an example of the problem. Consider the network in Figure 2-1, where the capacities are labeled along the edges. The source and the destination of the interdictor are nodes s and t. The interdictor has budget $\gamma = 2$. The user paths $P = \{p_1, p_2, p_3\}$ all have an initial flow value of 3. Let $\mathbf{f}$ be the s-t flow such that $\mathbf{f}(s, v_1) = \mathbf{f}(v_1, v_3) = \mathbf{f}(v_3, v_4) = \mathbf{f}(v_4, t) = 2$. In this example, the interdiciton strategy w such that $w(\mathbf{f}) = 1$ is optimal with $\Lambda(w, P) = 4$.

2.2.2 Computational Complexity

Before establishing the computational complexity, we first show some structural properties specific to the deterministic flow interdiction problem. Following from Proposition 1, there exists an interdiction strategy on the set of single-path flows that is optimal for the deterministic flow interdiction. We further extend this property, showing that there exists an optimal pure interdiction strategy.

Proposition 2. *For the deterministic flow interdiction problem, there exists an optimal (pure) interdiction strategy w such that $w(\mathbf{f}^*) = 1$ for some single-path flow $\mathbf{f}^*$.*

Proof. Building on proposition 1, let w be an optimal interdiction strategy that takes positive values only on single-path flows $\mathbf{q}_1, \ldots, \mathbf{q}_r$. Let $\mathbf{q}^* \in \arg\max_i \Lambda(\mathbf{q}_i, P)$. Consider the pure strategy w' with $w'(\mathbf{q}^*) = 1$. It follows that $\Lambda(w', P) = \Lambda(\mathbf{q}^*, P) \geq \sum_i w(\mathbf{q}_i)\Lambda(\mathbf{q}_i, P) = \Lambda(w, P)$, which proves the existence of an optimal pure strategy. □

From the proof of Propositions 1 and 2, we can obtain the following corollary.

Corollary 1. *Given an optimal strategy w to the deterministic flow interdiction problem, we can obtain another optimal strategy w' with $w'(\mathbf{f}^*) = 1$ for some single-path flow $\mathbf{f}^*$.*

Proof. For any $\mathbf{f}$ that $w(\mathbf{f}) > 0$, it can be decomposed into single-path flows $\mathbf{q}_1, \dots, \mathbf{q}_r$. From the proofs of Propositions 1 and 2, it follows that strategies $w_i, i \in \{1, \dots, r\}$ with $w_i(\frac{\gamma}{val(\mathbf{q}_i)}\mathbf{q}_i) = 1$ are all optimal. □

Corollary 1 states that a single path flow that maximizes $\Lambda(\cdot, P)$ can be obtained from an optimal interdiction strategy for the deterministic flow interdiciton problem in polynomial time. Hence, the NP-hardness of finding an optimal single-path flow implies the NP-hardness of the deterministic flow interdiction. Based on this result, we prove the NP-hardness of the deterministic flow interdiction problems.

Proposition 3. *The deterministic flow interdiction problem is NP-hard.*

Proof. The proof is done by reduction from the 3-satisfiability problem, which is a classical NP-Complete problem [45]. See Appendix 2.8 for the details. □

Remark: From the proof of Proposition 3, we have that even when the user paths are disjoint, the deterministic problem is still NP-hard.

2.2.3 Approximation Algorithm

Before presenting the algorithm, we extend some previous definitions. For any subset of edges $A \subseteq E$, imagine that the interdictor can interdict the edges in A by reducing their capacities by γ. We extend the definition of $\Lambda(\cdot, P)$ to A as $\Lambda(A, P) = \sum_i \lambda_i - T(A, P)$, where $T(A, P)$ is the optimal value of the maximization problem (2.1) with $\tilde{C}_A(e) = C(e) - \gamma \cdot \mathbb{1}_{\{e \in A\}}$. This provides an interpretation of $\Lambda(\cdot, P)$ as a set function on all subsets of E. Note that each single-path flow $\mathbf{f}$ can be equivalently represented as a set of edges $E_{\mathbf{f}}$ with $\mathbf{f}(e) = \gamma$ if and only if $e \in E_{\mathbf{f}}$. It follows that $\Lambda(\mathbf{f}, P) = \Lambda(E_{\mathbf{f}}, P)$, which links the definition of $\Lambda(\cdot, P)$ on single-path flows to that on sets of edges.

Our algorithm works on the optimization problem below.

$$\begin{aligned} \text{maximize} \quad & \Lambda(E_{\mathbf{f}}, P) \\ \textbf{s.t.} \quad & E_{\mathbf{f}} \text{ forms an } s\text{-}t \text{ path.} \end{aligned} \tag{2.4}$$

Let $E_{\mathbf{f}^*}$ be the optimal solution to (2.4) and $\mathbf{f}^*$ be its corresponding single-path flow. By Proposition 1, the strategy w with $w(\mathbf{f}^*)$ is an optimal interdiction strategy, and $\Lambda(E_{\mathbf{f}^*}, P) = \Lambda(w, P)$. Therefore, through approximating problem (2.4), our algorithm translates to an approximation to the deterministic flow interdiction problem. In the sequel, to better present the main idea of our algorithm, we first discuss the case where the user paths are edge-disjoint. After that, we generalize the results to the non-disjoint case.

C.1) Disjoint User Paths: When the user paths are edge-disjoint, for some interdicted edges $A \subseteq E$ and user paths $\{p_1, \ldots, p_k\}$ with initial values $\{\lambda_1, \ldots, \lambda_k\}$, the optimal solution to the max-flow problem (2.1) can be easily obtained as $\tilde{\lambda}_{iA} = \min\left(\lambda_i, \min_{e \in p_i} \tilde{C}_A(e)\right)$ for all i. It follows that the throughput reduction can be written as the sum of the throughput reduction on each paths, i.e., $\Lambda(A, P) = \sum_i (\lambda_i - \tilde{\lambda}_{iA})$. Based on this, we reason below that the set function $\Lambda(\cdot, P)$ has two important properties: *monotonicity and submodularity.*

Lemma 1. *Consider $\Lambda(\cdot, P) : 2^E \mapsto \mathbf{R}^*$ as a set function. Λ is:*

1. *Monotone: $\Lambda(A, P) \leq \Lambda(B, P)$ for all $A \subseteq B$;*

2. *Submodular: for all $A, B \subseteq E, e \in E$, if $A \subseteq B$, then $\Lambda(A \cup \{e\}, P) - \Lambda(A, P) \geq \Lambda(B \cup \{e\}, P) - \Lambda(B, P)$.*

Proof. The monotonicity is easily seen from the definition of Λ. The proof of submodularity is also straightforward. Note that for each user path i,

$$\begin{aligned}\lambda_i - \tilde{\lambda}_{iA} &= \lambda_i - \min(\lambda_i, \min_{e \in p_i}\{C(e) - \mathbb{1}_{\{e \in A\}}\}) \\ &= \lambda_i + \max(-\lambda_i, \max_{e \in p_i}\{-C(e) + \mathbb{1}_{\{e \in A\}}\})\end{aligned}$$

Since constant and linear functions are submodular, and the maximum of a set of submodular functions is also submodular, it follows that $\Lambda(A, P) = \sum_i (\lambda_i - \tilde{\lambda}_{iA})$ is submodular. □

Intuitively, an s-t path with large throughput reduction should have many intersections with different user paths. This intuition, combined with the monotonicity and submodularity of Λ, may suggest an efficient greedy approach to the optimization problem (2.4) that iteratively selects the edge with the maximum marginal gain with respect to Λ while sharing some s-t path with the edges that have already been selected. However, this is not the whole picture since such greedy selection might get stuck in some short s-t path and lose the chance of further including the edges that contribute to the throughput reduction. The latter aspect indicates the necessity of extensive search over the set of all s-t paths, but the number of s-t paths grows exponentially with n. Therefore, an algorithm with good performance guarantee and low time complexity must strike a balance between greedy optimization that harnesses the properties of Λ, and extensive search that avoids prematurely committing to some short path. The algorithm we propose, named as the Recursive Greedy algorithm, achieves such balance. It is based on the idea of [38]. The details of the algorithm are presented in **Algorithm 1** . In the description and analysis of the algorithm, $\Lambda_X(A, P) = \Lambda(A \cup X, P) - \Lambda(X, P)$ for $X, A \subseteq E$ represents the marginal gain of set A with respect to X. We use log to denote the logarithm with base two. For two nodes $u_1, u_2 \in V$, the shortest u_1-u_2 path is defined as the u_1-u_2 path with the smallest number of edges.

The recursive function *RG* lies at the heart of the Recursive Greedy algorithm. *RG* takes four parameters: source u_1, destination u_2, constructed subpath X and recursion depth i. It constructs a path from u_1 to u_2 that has a large value of $\Lambda_X(\cdot, P)$

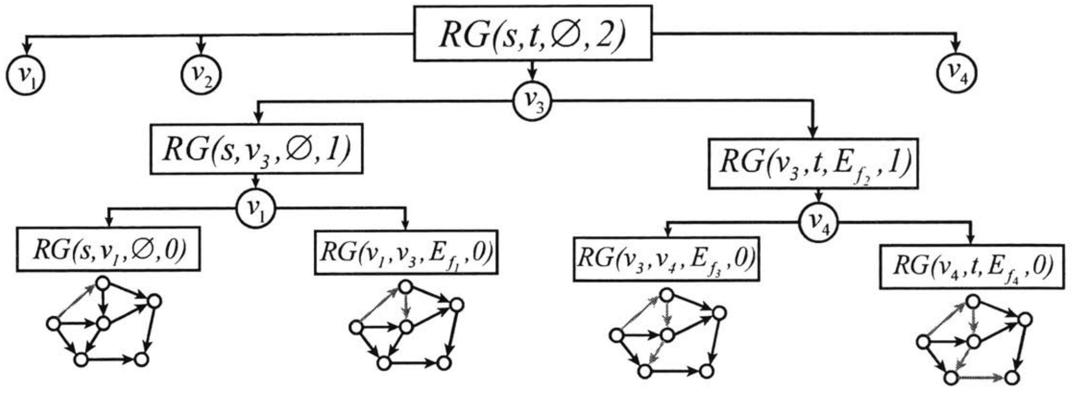


Figure 2-2: Illustration of the Recursive Greedy algorithm (with some intermediate steps omitted) on the example of Figure 2-1, where $E_{\mathbf{f}_1}, \ldots, E_{\mathbf{f}_4}$ are used to denote the sub-paths constructed during the recursion for ease of notation.

by recursively searching for a sequence of good anchors and greedily concatenating the sub-paths between anchors. The base case of the recursion is when the depth i reaches zero, then RG returns the shortest path between u_1 and u_2 if there exists one (step 2). Otherwise, it goes over all the nodes v in V (step 8), using v as an anchor to divide the search into two parts. For each v, it first calls a sub-procedure to search for sub-path from u_1 to v that maximizes $\Lambda_X(\cdot, P)$, with i decremented by 1 (step 9). After the first sub-procedure returns $E_{\mathbf{f}_1}$, it calls a second sub-procedure for sub-paths from v to u_2 (step 10). Note that the second sub-procedure is performed on the basis of the result of the first one, which reflects the greedy aspect of the algorithm. The two sub-paths concatenated serve as the u_1-u_2 path that RG obtains for anchor v. Finally, RG returns the path that maximizes $\Lambda_X(\cdot, P)$ over the ones that it has examined over all anchors (steps 11, 12 and 13).

The Recursive Greedy algorithm starts by invoking $RG(s, t, \emptyset, I)$ with I as the initial recursion depth. In the following, we show that the algorithm achieves a desirable performance guarantee as long as I is greater than certain threshold. An illustration of the algorithm with $I = 2$ on the previous example is shown in Figure 2-2. The optimal solution is returned by the path with anchors v_1, v_3, v_4.

Theorem 1. *If $I \geq \lceil \log d \rceil$, the Recursive Greedy algorithm returns an s-t path $E_{\mathbf{f}}$ with $\Lambda(E_{\mathbf{f}}, P) \geq \frac{1}{\lceil \log d \rceil + 1} \Lambda(E_{\mathbf{f}^*}, P)$, where d is the length of $E_{\mathbf{f}^*}$.*

Algorithm 1 The Recursive Greedy Algorithm

```
Input: Network graph $G(V, E)$, user paths $P = \{p_1, \dots, p_2\}$ with initial flow values
    $\{f_1, \dots, f_k\}$, Interdictor's source $s$, destination $t$ and budget $\gamma$
Output: The optimal $s$-$t$ path $E_\mathrm{f}$
 1: Run: $RG(s, t, \emptyset, I)$
    The Recursive Function $RG(u_1, u_2, X, i)$:
 2: $E_\mathrm{f} :=$ shortest $u_1$-$u_2$ path.
 3: if $E_\mathrm{f}$ does not exist then
 4:     return Infeasible
 5: if $i = 0$ then
 6:     return $E_\mathrm{f}$
 7: $r := \Lambda_X(E_\mathrm{f}, P)$.
 8: for all $v \in V$ do
 9:     $E_{\mathrm{f}_1} := RG(u_1, v, X, i-1)$.
10:     $E_{\mathrm{f}_2} := RG(v, u_2, X \cup E_{\mathrm{f}_1}, i-1)$.
11:     if $\Lambda_X(E_{\mathrm{f}_1} \cup E_{\mathrm{f}_2}, P) > r$ then
12:         $r := \Lambda_X(E_{\mathrm{f}_1} \cup E_{\mathrm{f}_2}, P)$, $E_\mathrm{f} := E_{\mathrm{f}_1} \cup E_{\mathrm{f}_2}$.
13: return $E_\mathrm{f}$
```

Proof. We prove a more general claim, that for all $u_1, u_2 \in V$, $X \subseteq E$, if $I \geq \lceil \log d \rceil$, the procedure $RG(u_1, u_2, X, I)$ returns an u_1-u_2 path E_f with $\Lambda_X(E_\mathrm{f}, P) \geq \frac{1}{\lceil \log d \rceil + 1} \Lambda_X(E_{\mathrm{f}^*}, P)$, where E_{f^*} is the u_1-u_2 path that maximizes $\Lambda(\cdot, P)$ and d is the length of E_{f^*}. The theorem follows from the claim by setting $u_1 = s$, $u_2 = t$ and $X = \emptyset$.

Let the nodes on the path E_{f^*} be $\{u_1 = v_0, \dots, v_d = u_2\}$. The proof is done by induction on d. First, for the base step, when $d = 1$, it means that there exists an edge between u_1 and u_2, which must be the shortest u_1-u_2 path. Obviously the procedure examines this path at step 2, and the claim follows. Next, suppose the claim holds for $d \leq l$. When $d = l + 1$, $I \geq 1$. Let $v^* = v_{\lceil \frac{d}{2} \rceil}$ and $E_{\mathrm{f}_1^*}, E_{\mathrm{f}_2^*}$ be the subpaths of E_{f^*} from u_1 to v^* and v^* to t, respectively. When RG uses v^* as an anchor, it first invokes $RG(u_1, v^*, X, I-1)$ that returns E_{f_1} and then invokes $RG(v^*, u_2, X \cup E_{\mathrm{f}_1}, I-1)$ that returns E_{f_2}. Let $E'_\mathrm{f} = E_{\mathrm{f}_1} \cup E_{\mathrm{f}_2}$. Our goal is to show that

$$\Lambda_X(E'_\mathrm{f}, P) \geq \frac{1}{\lceil \log d \rceil + 1} \Lambda_X\left(E_{\mathrm{f}^*}, P\right), \tag{2.5}$$

which proves the induction step, since the path E_f that $RG(u_1, u_2, X, I)$ returns must satisfy $\Lambda_X(E_{\mathsf{f}}, P) \geq \Lambda_X(E'_{\mathsf{f}}, P)$.

Since $I \geq \lceil \log d \rceil$, we have $I - 1 \geq \lceil \log d \rceil - 1 = \lceil \log \frac{d}{2} \rceil = \lceil \log \lceil \frac{d}{2} \rceil \rceil$. As $E_{\mathsf{f}_1^*}$ is a path of length $\lceil d/2 \rceil$ from u_1 to v^* and $E_{\mathsf{f}_2^*}$ is a path of length $\lfloor d/2 \rfloor$ from v^* to u_2, by the induction hypothesis,

$$\begin{aligned} \Lambda_X(E_{\mathsf{f}_1}, P) &\geq \frac{1}{\lceil \log d \rceil} \Lambda_X(E_{\mathsf{f}_1^*}, P), \\ \Lambda_{X \cup E_{\mathsf{f}_1}}(E_{\mathsf{f}_2}, P) &\geq \frac{1}{\lceil \log d \rceil} \Lambda_{X \cup E_{\mathsf{f}_1}}(E_{\mathsf{f}_2^*}, P). \end{aligned}$$

By the submodularity of Λ (Lemma 1), we have

$$\begin{aligned} \Lambda_X(E_{\mathsf{f}_1^*}, P) &\geq \Lambda_{X \cup E'_{\mathsf{f}}}(E_{\mathsf{f}_1^*}, P) \\ \Lambda_{X \cup E_{\mathsf{f}_1}}(E_{\mathsf{f}_2^*}, P) &\geq \Lambda_{X \cup E'_{\mathsf{f}}}(E_{\mathsf{f}_2^*}, P) \end{aligned}$$

Using this, we sum the two inequalities obtained form the induction hypothesis and get

$$\begin{aligned} &\Lambda_X(E'_{\mathsf{f}}, P) \geq \frac{1}{\lceil \log d \rceil} \Big(\Lambda_X(E_{\mathsf{f}_1^*}, P) + \Lambda_{X \cup E_{\mathsf{f}_1}}(E_{\mathsf{f}_2^*}, P) \Big) \\ &\geq \frac{1}{\lceil \log d \rceil} \Big(\Lambda_{X \cup E'_{\mathsf{f}}}(E_{\mathsf{f}_1^*}, P) + \Lambda_{X \cup E'_{\mathsf{f}}}(E_{\mathsf{f}_2^*}, P) \Big). \end{aligned}$$

Again, by Lemma 1, we have,

$$\Lambda_{X \cup E'_{\mathsf{f}}}(E_{\mathsf{f}_2^*}, P) \geq \Lambda_{X \cup E'_{\mathsf{f}} \cup E_{\mathsf{f}_1^*}}(E_{\mathsf{f}_2^*}, P)$$

It follows that

$$\begin{aligned} &\Lambda_X(E'_{\mathsf{f}}, P) \geq \frac{1}{\lceil \log d \rceil} \Big(\Lambda_{X \cup E'_{\mathsf{f}}}(E_{\mathsf{f}_1^*}, P) + \Lambda_{X \cup E'_{\mathsf{f}} \cup E_{\mathsf{f}_1^*}}(E_{\mathsf{f}_2^*}, P) \Big) \\ &= \frac{1}{\lceil \log d \rceil} \Big(\Lambda\left(X \cup E'_{\mathsf{f}} \cup E_{\mathsf{f}^*}, P\right) - \Lambda\left(X \cup E'_{\mathsf{f}}, P\right) \Big) && (2.6) \\ &\geq \frac{1}{\lceil \log d \rceil} \Big(\Lambda\left(X \cup E_{\mathsf{f}^*}, P\right) - \Lambda\left(X \cup E'_{\mathsf{f}}, P\right) \Big) && (2.7) \\ &= \frac{1}{\lceil \log d \rceil} \Big(\Lambda_X\left(E_{\mathsf{f}^*}, P\right) - \Lambda_X\left(E'_{\mathsf{f}}, P\right) \Big), && (2.8) \end{aligned}$$

where equality (2.6) follows from the definition of Λ_X and that $E_{\mathbf{f}^*} = E_{\mathbf{f}_1^*} \cup E_{\mathbf{f}_2^*}$, inequality (2.7) follows from the monotonicity of Λ and equality (2.8) follows also from the definition Λ_X. From (2.8), we obtain (2.5), which concludes the proof. $\square$

Time Complexity: The bound on the Recursive Greedy algorithm's running time is easy to establish. As we invoke at most $2n$ sub-procedures at each level of recursion and the computation of Λ takes $O(m)$ time, the time complexity of the algorithm is $O((2n)^I m)$. Taking $I = \log n \geq \lceil \log d \rceil$,[1] we get an algorithm with a logarithmic approximation ratio of $1/(\lceil \log d \rceil + 1)$ with a quasi-polynomial time complexity of $O((2n)^{\log n} m)$.

Remark: First, note that the proof of Theorem 1 only relies on the monotonicity and submodularity of Λ. Therefore, the Recursive Greedy algorithm works for any monotone and submodular function on the subsets of E. Second, we can generalize **Algorithm 1** to one that uses more than one anchors at step 8. The generalization is given in Appendix 2.8.1. When the algorithm uses $a - 1$ anchors, it achieves an approximation ratio of $1/(\lceil \log_a d \rceil + 1)$ in $O((an)^{(a-1)\log_a n} m)$ time. The parameter a can thus control the tradeoff between the performance guarantee and the time complexity of the algorithm.

C.2) Non-disjoint User Paths: When the user paths are not disjoint, the problem becomes more challenging. First, notice that $\tilde{\lambda}_{iA} = \min\left(\lambda_i, \min_{e \in p_i} \tilde{C}_A(e)\right)$ no longer holds due to the constraints in (2.2) that couple different $\tilde{\lambda}_i$'s together. More importantly, Λ actually loses the submodular property when the user paths are not disjoint, which prevents the direct application of the Recursive Greedy algorithm. We tackle the issues through approximating Λ with a monotone and submodular function $\bar{\Lambda}$, and run the Recursive Greedy algorithm on $\bar{\Lambda}$. The performance guarantee of the algorithm can be obtained by bounding the gap between Λ and $\bar{\Lambda}$.

Let $E_0 \subseteq E$ be the set of edges that belong to some user path. This is also the set of edges that appear in constraints (2.2). We partition E_0 into two sets E_1 and E_2, where E_1 is the set of edges that belong to only one user path, and E_2 is the

[1]Strictly speaking, we need to set $I = \lceil \log n \rceil$. We omit the ceiling function here for ease of notations.

set of edges that belong to at least two (intersecting) user paths. Following this, we define $\bar{\Lambda}(A, P), A \subseteq E$ to be evaluated through the two-phase procedure below. The procedure first goes edges in E_1 (Phase I), setting

$$\tilde{\lambda}_{iA}^{(1)} := \min\left(\lambda_i, \min_{e \in p_i, e \in E_1} \{\tilde{C}_A(e)\}\right), \quad \forall i.$$

Then, it goes over edges in E_2 (Phase II), setting

$$\tilde{\lambda}_{iA}^{(2)} := \tilde{\lambda}_{iA}^{(1)} \cdot \prod_{e \in p_i, e \in E_2, \tilde{C}_A(e) \leq \sum_{p_j \ni e} \lambda_j} \frac{\tilde{C}_A(e)}{\sum_{p_j \ni e} \lambda_j}, \quad \forall i.$$

Finally, it sets $\bar{\Lambda}(A, P) = \sum_i \lambda_i - \sum_i \tilde{\lambda}_{iA}^{(2)}$.

The procedure uses $\{\tilde{\lambda}_{iA}^{(2)}\}$, a set of flow values on user paths, as an approximate solution to the max-flow problem (2.1). The solution is obtained through first setting the flow values to $\{\lambda_i\}$ and then gradually decreasing them until the constraints are satisfied. In Phase I, the flow values are decreased to satisfy the capacity constraints posed by edge in E_1. In Phase II, the flow values are further reduced to compensate for the capacity violations on edges in E_2 through multiplying a factor $\frac{\tilde{C}_A(e)}{\sum_{p_j \ni e} \lambda_j}$, which is equal to the ratio between the capacity of e after the interdiction and the sum of flow values on e before the interdiction, to the flow value of each user path containing e, for each $e \in E_2$. Typically, Phase II overcompensates and thus $\bar{\Lambda}$ is an upper bound of Λ. But as we will show, the gap between $\bar{\Lambda}$ and Λ is moderate and such overcompensation guarantees the submodularity of $\bar{\Lambda}$.

Substituting Λ with $\bar{\Lambda}$ in **Algorithm 1**, we obtain the Recursive Greedy algorithm for the case of non-disjoint user paths. We will refer to this algorithm as the **Extended Recursive Greedy algorithm**. The name is justified by noting that when the user paths are disjoint, $E_2 = \emptyset$ and $\bar{\Lambda} = \Lambda$, the Extended Recursive Greedy algorithm degenerates to **Algorithm 1**.

Before analyzing the performance of the algorithm, we establish two lemmas that show the monotonicity and submodularity of $\bar{\Lambda}$, and bound the gap between $\bar{\Lambda}$ and Λ, respectively. The proofs of the lemmas are given in Appendix 2.9

Lemma 2. *Consider $\bar{\Lambda}(\cdot, P) : 2^E \mapsto \mathbb{R}^*$ as a set function. $\bar{\Lambda}(\cdot, P)$ is monotone and submodular.*

Lemma 3. $\Lambda(A,P) \leq \bar{\Lambda}(A,P) \leq (b+1)\cdot\Lambda(A,P)$ *for all $A \subseteq E$, where $b = \max_i |E_2 \cap p_i|$,*[2] *i.e., the maximum number of edges that a user path shares with other user paths.*

Now, we are ready to analyze the performance of the Extended Recursive Greedy algorithm.

Theorem 2. *If $I \geq \lceil \log d \rceil$, then the Extended Recursive Greedy algorithm returns an s-t path E_{f} that satisfies*

$$\Lambda(E_{\mathrm{f}}, P) \geq \frac{1}{(b+1)\cdot(\lceil \log d \rceil + 1)} \Lambda(E_{\mathrm{f}^*}, P),$$

where d is the length of E_{f^*} and $b = \max_i |E_2 \cap p_i|$.

Proof. By Lemma 2 and Theorem 1, we have $\bar{\Lambda}(E_{\mathrm{f}}, P) \geq \frac{1}{(\lceil \log d \rceil + 1)} \bar{\Lambda}(E_{\mathrm{f}^*}, P)$ when $I \geq \lceil \log d \rceil$. Invoking Lemma 3, we obtain that

$$\begin{aligned}\Lambda(E_{\mathrm{f}}, P) &\geq \frac{1}{b+1}\bar{\Lambda}(E_{\mathrm{f}}, P) \geq \frac{1}{(b+1)\cdot(\lceil \log d \rceil + 1)}\bar{\Lambda}(E_{\mathrm{f}^*}, P) \\ &\geq \frac{1}{(b+1)\cdot(\lceil \log d \rceil + 1)}\Lambda(E_{\mathrm{f}^*}, P),\end{aligned}$$

which concludes the proof. □

Note that the computation of $\bar{\Lambda}$ takes $O(m)$ time. Therefore, taking $I = \log n$ we get a $\frac{1}{(b+1)\cdot(\lceil \log d \rceil + 1)}$-approximation algorithm with a time complexity of $O((2n)^{\log n} m)$. Although in the worst case, b can be at the same order as n. In the cases, b is of $O(\log n)$, and the Extended Recursive Greedy algorithm still enjoys a logarithmic approximation ratio.

[2] $|A|$ denotes the cardinality of set A

2.3 Robust Flow Interdiction

In this section, we investigate the robust flow interdiction problem. Following the road map of deterministic flow interdiction, we first describe the formal definition of the problem, then show its computational complexity, and finally present the approximation framework for the problem.

2.3.1 Problem Formulation

While deterministic flow interdiction considers the case where the interdictor has definitive knowledge of the user paths, robust flow interdiction concerns scenarios where such knowledge is not available. We model this more complicated situation using the robust optimization framework [37]. Instead of having certain knowledge of P, the interdictor only knows that P lies in an uncertainty set $\mathcal{U} = \{P_1, \ldots, P_\xi\}$. Each $P_l = \{p_{l1}, \ldots, p_{lk_l}\} \in \mathcal{U}$, associated with initial flow values $\{\lambda_{l1}, \ldots, \lambda_{lk_l}\}$, is a candidate set of paths that the users are operating on. The interdictor aims to hedge against the worst case, maximizing the minimum throughput reduction achieved over all candidates P.

Definition 3 (Robust Flow Interdiction). *Given the uncertain set $\mathcal{U} = \{P_1, \ldots, P_\xi\}$ of the user paths and the associated initial flow values on user paths for each $P \in \mathcal{U}$, the robust flow interdiction problem seeks an interdiction strategy w that maximizes the worst case throughput reduction, i.e., $w \in \arg\max_{w'} \min_{P \in \mathcal{U}} \Lambda(w', P)$.*

Example: As an example of the robust flow interdiction problem, we again consider the network in Figure 2-1. The interdictor has source s, detination t and budget $\gamma = 2$. Assume that the interdictor only knows that the users are sending flow on either $\{p_1, p_2\}$ or $\{p_1, p_3\}$, and the initial flow values on p_1, p_2, p_3 are all three. This corresponds to the robust flow interdiction with $\mathcal{U} = \{\{p_1, p_2\}, \{p_1, p_3\}\}$. Let $\mathbf{f}_1$ be the s-t flow such that $\mathbf{f}_1(s, v_3) = \mathbf{f}_1(v_3, v_4) = \mathbf{f}_1(v_4, t) = 2$ and $\mathbf{f}_2$ be the s-t flow such that $\mathbf{f}_2(s, v_1) = \mathbf{f}_2(v_1, v_3) = \mathbf{f}_2(v_3, v_2) = \mathbf{f}_2(v_2, t) = 2$. The optimal strategy in this case is $w(\mathbf{f}_1) = 1/3, w(\mathbf{f}_2) = 2/3$, and the worst case throughput reduction equals $8/3$ as

$\Lambda(w, \{p_1, p_2\}) = \Lambda(w, \{p_1, p_3\}) = 8/3$. Note that in this example, no pure interdiction strategy can achieve a worst case throughput reduction of 8/3, which demonstrates the superiority of mixed strategies in the robust flow interdiction setting.

The robust flow interdiction problem subsumes the deterministic one as a special case by setting $\mathcal{U} = \{P\}$. Therefore, we immediately have the following proposition.

Proposition 4. *The robust flow interdiction problem is NP-hard.*

Before presenting our approximation framework, we present a linear programming (LP) formulation that serves as an alternative solution to the robust flow interdiction problem. According to Proposition 1, we can restrict our attention to distributions on the set of single-path flows with value γ. Therefore, in the following, *the distributions we refer to are all on the set of single-path flows in $\mathcal{F}_\gamma$.* We enumerate such single-path flows in an arbitrary order and associate with each single-path flow $\mathbf{f}_i$ a variable w_i. Consider the linear program:

$$\begin{aligned} &\text{maximize } z \\ &\textbf{s.t.} \ \sum_i w_i \Lambda(\mathbf{f}_i, P) \geq z, \quad \forall P \in \mathcal{U} \\ &\qquad \sum_i w_i = 1 \\ &\qquad w_i \geq 0, \quad \forall i \end{aligned} \tag{2.9}$$

Clearly, the solution to the LP corresponds to an optimal interdiction strategy w to the robust flow interdiction problem with $w(\mathbf{f}_i) = w_i$. Hence, formulating and solving the LP is a natural algorithm for the robust flow interdiction. However, as the number of single-path flows can be exponential in the number of nodes n, the LP may contain an exponential number of variables. It follows that the algorithm has an undesirable exponential time complexity. We use this algorithm in the simulations to obtain optimal interdiction strategies for comparisons with our approximation framework. Another issue arises when the number of single-path flows is exponential in the number of nodes, that is, even outputting the strategy w takes exponential time. This makes it impractical and unfair to compare any sub-exponential time approximation procedure to the optimal solution. We get around this issue by comparing our solution to the optimal interdiction strategy that takes non-zero values on at most N_0 single-path flows, where N_0 is a pre-specified number bounded by some polynomial of n. We refer to such strategies as N_0-bounded strategies. The optimal N_0-bounded strategy corresponds to the best strategy that uses at most N_0 different interdicting flows. Note that such restriction does not trivialize the problem since we place no limitation on the set but just the number of single-path flows that the interdictor can use.

2.3.2 Approximation Framework

In this section, we present the approximation framework we propose for the robust flow interdiction problem. As a generalization of the deterministic version, the robust flow interdiction is more complicated since it involves maximizing the minimum of a set of functions. The (Extended) Recursive Greedy algorithm cannot be directly adapted to this case. Instead, we design an approximation framework that integrates the Extended Recursive Greedy algorithm as a sub-procedure. The framework only incurs a logarithmic loss in terms of approximation ratio.

The description and analysis of the approximation framework are carried out in three steps. First, we justify that it is sufficient to consider the robust flow interdiction problem with parameters taking rational/integral values. In the second step, building on the rationality/integrality of parameter values, we convert the problem to a sequence of integer linear programs. Finally, we solve the sequence of integer programs through iteratively invoking the Extended Recursive Greedy algorithm.

Rationalizing the Parameters

In the first step, we show that not much is lost if we only consider the interdiction strategies that take rational values and restrict the throughput reduction to take integer values. Specifically, let $N = N_0^2 + N_0$ and $\mathbb{Q}_N = \{\frac{\beta}{N} : \beta \in \mathbb{N}, 0 \leq \beta \leq N\}$ be the set of non-negative rational numbers that can be represented with N as denominator. Further, we define $\mathcal{W}_N$ to be the set of strategies that take value in $\mathbb{Q}_N$, i.e., $\mathcal{W}_N = \{w : \mathcal{F}_\gamma \mapsto \mathbb{Q}_N, \sum_{\mathbf{f}} w(\mathbf{f}) = 1\}$. We use w^* to represent the optimal N_0-bounded interdiction strategy, and w_N^* to represent optimal strategy in $\mathcal{W}_N$. The following lemma states that w^* can be well approximated by w_N^*.

Lemma 4. *For all $P \in \mathcal{U}$, $\Lambda(w_N^*, P) \geq \frac{N_0}{N_0+1}\Lambda(w^*, P)$.*

Proof. Consider $\tilde{w}^*$ such that $\tilde{w}^*(\mathbf{f}) = \frac{\lceil N_0^2 w^*(\mathbf{f})\rceil}{N_0^2+N_0}$ for all $w^*(\mathbf{f}) > 0$ and $\tilde{w}^*(\mathbf{f}) = 0$ otherwise. Since w^* is a N_0-bounded strategy, $\sum_{\mathbf{f}} \tilde{w}^*(\mathbf{f}) \leq \frac{N_0+N_0^2\sum_{\mathbf{f}} w^*(\mathbf{f})}{N_0^2+N_0} = 1$. Hence, we can augment $\tilde{w}^*$ into a strategy in $\mathcal{W}_N$ by adding $1 - \sum_{\mathbf{f}'} \tilde{w}^*(\mathbf{f}')$ to some $\tilde{w}^*(\mathbf{f})$. With a little abuse of notation, we use $\tilde{w}^*$ to denote the resulting strategy. By the definition of $\tilde{w}^*$, we have

$$\sum_{\mathbf{f}} \tilde{w}^*(\mathbf{f})\Lambda(\mathbf{f}, P) \geq \frac{N_0^2}{N_0^2+N_0}\sum_{\mathbf{f}} w^*(\mathbf{f})\Lambda(\mathbf{f}, P) = \frac{N_0}{N_0+1}\Lambda(w^*, P).$$

As $\tilde{w}^* \in \mathcal{W}_N$, it follows that $\Lambda(w_N^*, P) \geq \Lambda(\tilde{w}^*, P) = \sum_{\mathbf{f}} \tilde{w}^*(\mathbf{f})\Lambda(\mathbf{f}, P) \geq \frac{N_0}{N_0+1}\Lambda(w^*, P)$. □

We now proceed to argue that it suffices to consider the throughput reduction function Λ to take integral values that are bounded by some polynomial of n. First, when the integrality of Λ is not satisfied, we can always use standard scaling and rounding tricks to get a new instance of the problem, where Λ takes integral values. Our framework can be applied to the new instance, yielding an interdiction strategy that has almost the same performance guarantee for both the original and the new instances. We defer the formal statement and proof of this to Appendix 2.9.2, as it involves definitions in subsequent sections. Second, since γ is bounded by some polynomial of n, $\max_{w,P} \Lambda(w, P)$ is also bounded by some polynomial of n. Now, let $M = N \max_{w,P} \Lambda(w, P)$. We can thus without loss of generality assume that M is an integer bounded by some polynomial of n.

With the above results, we move into the second step, that converts the robust flow interdiction problem into a sequence of integer linear programs.

Converting into Integer Linear Programs

Recall the enumeration of single-path flows in the LP (2.9). This time, we associate each flow $\mathbf{f}_i$ with an integral variable x_i. Consider the following integer program $ILP(\kappa)$ parameterized by a positive integer $\kappa \leq M$.

$$\text{minimize } \textstyle\sum_i x_i \tag{2.10}$$

$$\textbf{s.t. } \sum_i x_i \Lambda(\mathbf{f}_i, P) \geq \kappa, \quad \forall P \in \mathcal{U} \tag{2.11}$$

$$x_i \in \mathbb{N}, \quad \forall i \tag{2.12}$$

Each x_i indicates the number of times $\mathbf{f}_i$ is selected. $ILP(\kappa)$ can be interpreted as selecting the single-path flows for the minimum total number of times that achieve a throughput reduction of κ for all candidate P.

For each κ, we denote by N_κ the optimal value of $ILP(\kappa)$. If we can obtain an optimal solution $\{x\}$ to $ILP(\kappa)$, then the strategy w with $w(\mathbf{f}_i) = x_i/N_\kappa$ satisfies $\min_{P\in\mathcal{U}} \Lambda(w, P) \geq \kappa/N_\kappa$. In the following lemma, we show that the strategy constructed according to the solution to the integer program with the maximum value of κ/N_κ is a close approximation to the optimal N_0-bounded strategy in terms of worst case throughput reduction.

Lemma 5. *Let* $\kappa^* = \arg\max_{1\leq\kappa\leq M}(\kappa/N_\kappa)$. *We have* $\frac{\kappa^*}{N_{\kappa^*}} \geq \min_{P\in\mathcal{U}} \Lambda(w_N^*, P) \geq \frac{N_0}{N_0+1} \min_{P\in\mathcal{U}} \Lambda(w^*, P)$.

Proof. Define κ' to be $\min_{P\in\mathcal{U}} \sum_{\mathbf{f}} N w_N^*(\mathbf{f})\Lambda(\mathbf{f}, P) = \min_{P\in\mathcal{U}} N\Lambda(w_N^*, P)$. Note that κ' is a positive integer and $\kappa' \leq M$. Thus, by the definition of κ^*, we have $\kappa'/N_{\kappa'} \leq \kappa^*/N_{\kappa^*}$. Also, observe that the solution $\{x\}$ with $x_i = N w_N^*(\mathbf{f}_i)$ is feasible to $ILP(\kappa')$. Therefore, $N_{\kappa'} \leq \sum_i w_N^*(\mathbf{f}_i) N = N$. It follows that

$$\frac{\kappa^*}{N_{\kappa^*}} \geq \frac{\kappa'}{N_{\kappa'}} \geq \frac{\kappa'}{N} = \min_{P\in\mathcal{U}} \Lambda(w_N^*, P) \geq \frac{N_0}{N_0+1} \min_{P\in\mathcal{U}} \Lambda(w^*, P).$$

□

Connecting the analysis so far, we have a clear procedure to compute a near-optimal interdiction strategy for the robust flow interdiction. First, we construct and solve $ILP(\kappa)$ for $1 \leq \kappa \leq M$. Second, we take optimal solution with the maximal κ/N_κ and obtain its corresponding interdiction strategy, which is within a factor of $\frac{N_0}{N_0+1}$ to the optimal N_0-bounded strategy. The final step of our framework is devoted to solving $ILP(\kappa)$.

Solving the Integer Linear Programs

Resembling (2.9), each $ILP(\kappa)$ involves potentially exponential number of variables. What is different and important is that, we can obtain a $\frac{1}{\log M}$-approximation through a greedy scheme that iteratively chooses a single-path flow according to the following criterion: let $\{x\}$ indicate the collection of flows that have been chosen so far, i.e., each $\mathbf{f}_i$ has been chosen for x_i times. Let i^* be

$$\arg\max_i \sum_{P\in\mathcal{U},\kappa\geq\sum_j x_j\Lambda(\mathbf{f}_j,P)} \min\{\kappa - \sum_j x_j\Lambda(\mathbf{f}_j,P), \Lambda(\mathbf{f}_i,P)\}. \tag{2.13}$$

The greedy scheme chooses $\mathbf{f}_{i^*}$ at the current iteration and increments x_{i^*} by 1. The above procedure is repeated until we have $\sum_i x_i\Lambda(\mathbf{f}_i,P) \geq k$ for all $P \in \mathcal{U}$. Moreover, if we apply an α-approximate greedy scheme, which chooses $\mathbf{f}_i$ that is an α-optimal solution to (2.13), then the final solution we obtain is $\alpha \log M$-optimal. Essentially, (2.13) selects the flow that provides the maximum marginal gain with respect to satisfying the constraints (2.11) for all $P \in \mathcal{U}$. That the (α-approximate) greedy scheme achieves an logarithmic approximation follows from the relation of $ILP(\kappa)$ to the multiset-multicover problems and the results therein [39], which we omit here due to space limitation. Now recall the equivalence between $\Lambda(\mathbf{f},P)$ and $\Lambda(E_\mathbf{f},P)$ established in Section 2.2. We proceed to show that the Recursive Greedy algorithm can be used to construct an approximate greedy scheme. First, we have the following lemma.

Lemma 6. *If Λ is monotone and submodular, then the objective function of (2.13) is also monotone and submodular.*

Proof. Note that at any iteration, $\sum_j x_j\Lambda(\mathbf{f}_j,P)$ is a known constant. Hence, for each

P, $\min\{\kappa - \sum_j x_j \Lambda(\mathbf{f}_j, P), \Lambda(\mathbf{f}_i, P)\}$ is the minimum of a constant and a monotone submodular function, which is also monotone and submodular. It follows that the objective function of (2.13) is monotone and submodular. □

By Lemma 6, the Recursive Greedy algorithm (or the Extended Recursive Greedy algorithm using $\bar{\Lambda}$ instead of Λ when the user paths are not disjoint) can be applied to the maximization of (2.13) and enjoys the same performance guarantee as in Theorems 1 and 2. Hence, the final step can be completed by an approximate greedy scheme that iteratively invokes the (Extended) Recursive Greedy algorithm. We now summarize the three steps of our approximation framework for the robust flow interdiction as **Algorithm 2** and analyze its performance.

Algorithm 2 Algorithm for the Robust Flow Interdiction

```
Input: Network graph $G$, Uncertainty set $\mathcal{U} = \{P_1, \dots, P_\xi\}$, Interdictor's source $s$,
    destination $t$ and budget $\gamma$
Output: Interdiction Strategy $w$
 1: Formulate $ILP(\kappa)$ for $1 \le \kappa \le M$.
 2: Solve each $ILP(\kappa)$ using the approximate greedy scheme based on the (Extended)
    Recursive Greedy algorithm.
 3: Take the solution $\{x\}$ to $ILP(\kappa)$ with the maximum value of $\kappa / \sum_j x_j$ and con-
    struct $w$ by setting $w(\mathbf{f}_i) = x_i / \sum_j x_j$ for all $i$.
 4: return $w$
```

Theorem 3. ***Algorithm** 2 returns an interdiction strategy w that satisfies*

$$\begin{aligned}&\min_{P\in\mathcal{U}} \Lambda(w, P)\\ &\ge \left(\frac{N_0}{(N_0+1)(b+1)\log M \cdot (\lceil \log d \rceil + 1)}\right) \min_{P\in\mathcal{U}} \Lambda(w^*, P),\end{aligned}$$

where w^* is the optimal N_0-bounded strategy.

Proof. Let $ILP(\kappa)$ and $\{x\}$ be the integer linear program and its solution that correspond to w. We inherit the definition of κ^* in Lemma 5 and further define $\{x^*\}$ to be the solution that **Algorithm 2** computes for $ILP(\kappa^*)$. We have

$$\begin{aligned}
&\min_{P\in\mathcal{U}} \Lambda(w,P) = \min_{P\in\mathcal{U}} \sum_i w(\mathbf{f}_i)\Lambda(\mathbf{f}_i,P) \\
&= \min_{P\in\mathcal{U}} \sum_i \frac{x_i}{\sum_j x_j}\Lambda(\mathbf{f}_i,P) \geq \frac{\kappa}{\sum_j x_j} \geq \frac{\kappa^*}{\sum_j x_j^*} \\
&\geq \left(\frac{1}{(b+1)\log M \cdot (\lceil \log d \rceil + 1)}\right)\frac{\kappa^*}{N_{\kappa^*}} \qquad (2.14) \\
&\geq \left(\frac{N_0}{(N_0+1)(b+1)\log M \cdot (\lceil \log d \rceil + 1)}\right)\min_{P\in\mathcal{U}} \Lambda(w^*,P), \qquad (2.15)
\end{aligned}$$

where inequality (2.14) follows from Theorem 2 and the results in [39], and inequality (2.15) follows from Lemma 5. □

Time Complexity: Note that **Algorithm 2** solves M integer linear programs, and it takes at most $M\xi$ calls of the (Extended) Recursive Greedy algorithm for each program since the number of iterations is bounded by $M\xi$, where $\xi = |\mathcal{U}|$. Furthermore, at the third step, there are at most $N_\kappa \leq M\xi$ variables with non-zero values in the solution $\{x\}$, which implies that w can be output in $O(M\xi)$ time. Therefore, the time complexity of **Algorithm 2** is $O\left(m(M\xi)^2(2n)^{\log n}\right)$.

2.4 Simulations

In this section, we present our evaluation of the performance of the proposed algorithms. We first introduce the simulation environment in the following and then show the detailed results in subsequent sections.

2.4.1 Simulation Setting

We adopt the Gnutella peer to peer network data set from [43]. We extract 20 networks of 1000 nodes, and make the networks acyclic by removing a minimal feedback edge set from each of them. The capacities of the edges are sampled from a normal distribution with mean 20 and standard deviation 3. The budget of the interdictor is set to the minimum capacity of the edges in each network.

2.4.2 Deterministic Flow Interdiction

In the deterministic flow interdiction, we divide our simulations into two parts, where the user paths are disjoint and non-disjoint respectively. In the first part, we designate k disjoint paths in each network as user paths with k varying in $\{10, 20, \ldots, 100\}$. In the second part, we follow the similar route, except that the user paths are randomly chosen without guaranteeing their disjointness. For each network, we randomly select five connected node pairs as the source and destination of the interdictor. Thus, for each number of user paths, we have 100 simulation scenarios in total (20 networks times 5 s-t pairs).

Algorithms Involved in Performance Comparisons

We apply the Recursive Greedy algorithm when the user paths are disjoint and run the extended one when the user paths are non-disjoint. We vary the recursion depth, i.e., the value of I in **Algorithm 1** to evaluate its influence on the algorithms' performance. Our algorithms are compared to a *brute force* algorithm that enumerates all the paths between the interdictor's source and destination, which computes the optimal interdiction strategy.

Performance Metric

We calculate the ratio of the throughput reduction of the interdiction strategies by our algorithms to that of the optimal solutions obtained by the brute force algorithm. The results reported are the average over all the 100 scenarios.

Simulation Results

We plot the results of our algorithms on deterministic flow interdiction with disjoint and non-disjoint user paths in Figures 2-3(a) and 2-3(b).

From Figure 2-3(a), we can see that: (i). by setting the recursion depth to two, we get interdiction strategies with throughput reduction more than 90% of the optimal (0.9-approximation) and (ii). by setting the recursion depth to three, we recover the optimal interdiction strategies. Furthermore, in the simulations, we find that when the recursive depth is three, the number of paths examined by the Recursive Greedy algorithm is just about one fifth of the total number of s-t paths. This suggests that the typical performance and running time are even better than what the theoretical analysis predicts. Finally, we observe that, in general, our algorithms perform better when the number of user paths is large. This observation also holds in subsequent cases. One possible explanation for this is that more user paths present more opportunities for throughput reduction, making (near-)optimal interdicting flows easier to find.

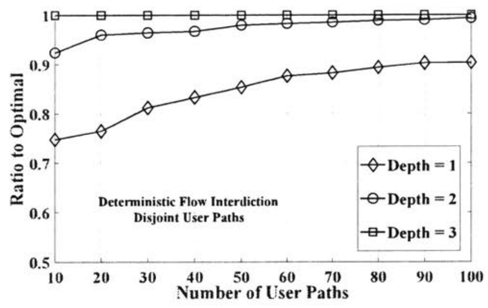
As demonstrated in Figure 2-3(b), the deterministic flow interdiction is harder to approximate when the user paths are non-disjoint. But we can still get 0.8-approximations with a recursion depth of three and 0.95-approximations with a recursion depth of four. Also, though we have not plotted in the figure, we have seen that increasing the recursion depth to five or six does not further improve the performance. Therefore, the gap between the Extended Recursive Greedy algorithm with depth of four and the optimal can be attributed to the loss brought by the approximate throughput reduction function $\bar{\Lambda}$.

2.4.3 Robust Flow Interdiction

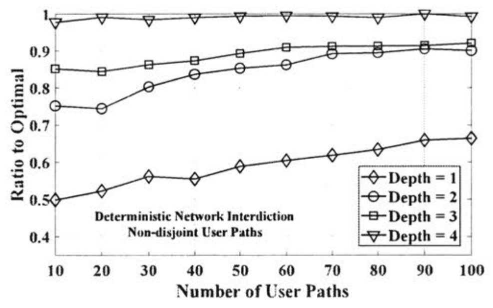
In the robust flow interdiction, we randomly select 10 groups of k paths as the uncertainty set $\mathcal{U}$ for $k \in \{10, 20, \ldots, 100\}$. Similar as before, we randomly selected 5 source-destination pairs for the interdictor in each network and form 100 scenarios for each k.

Algorithms Involved in Performance Comparisons

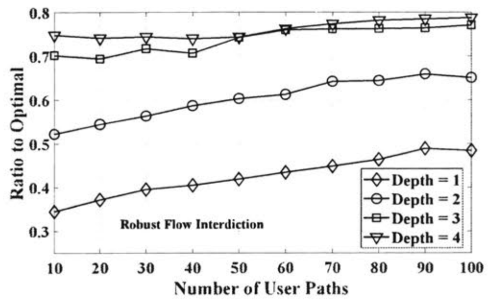
We embed the Extended Recursive Greedy algorithm with different recursion depths in our proposed approximation framework (**Algorithm 2**). The optimal solution in this case is obtained by solving the LP (2.9).



(a)



(b)



(c)

Figure 2-3: Ratio of throughput reduction of the solutions by our algorithms to the optimal.

Performance Metric

For all strategies w computed by our algorithms, we calculate the ratio of $\min_{P \in \mathcal{U}} \Lambda(w, P)$ to that of the optimal. The results reported are again averaged over all the scenarios.

Simulation Results

We plot the results in Figure 2-3(c). Taking the depth as four, our approximation framework achieves interdiction strategies that are more than 70% of the optimal (0.7-approximation). As in the previous case, we have implemented the framework with recursion depth of five and six but found that it did not improve the performance.

2.5 Discussion of General Networks

In this section, we extend our network interdiction paradigm and the two flow interdiction problems to general networks. Our network interdiction paradigm can be straightforwardly extended to general networks by allowing the network graph to be a general directed graph. One caveat is that we need to additionally restrict the flows that the interdictor injects to be free of cycles. Since otherwise, as the flow value of a cycle is zero, the interdictor would be able to consume the capacities of the edges in any cycle without spending any of its budget, which would lead to meaningless solutions. Under the generalized paradigm, the deterministic and robust flow interdiction problems can be defined in the same way as Definitions 2 and 3. For the network interdiction paradigm on general networks, Proposition 1 still holds. But the Extended Recursive Greedy algorithm will break down since the edge set it returns will be an s-t walk instead of an s-t path (i.e. it may contain cycles). Furthermore, we can prove by an approximation-preserving reduction from the Longest Path problem in directed graphs [44] that there is no polynomial time (quasi-polynomial time) algorithm with an approximation ratio of $O(n^{1-\delta})$ for any $\delta > 0$ unless $P = NP$ ($DTIME(O(n^{\log n})) = NP$).[3] The reduction works by defining the graph in the Longest Path problem instance as the network graph and designating each edge as a user path. We further set the capacities of the edges and the interdictor's budget as one. Thus, the optimal single-path flow would essentially be the longest path from the interdictor's source and destination, with the throughput reduction equaling the length of the path it corresponds to. Enumerating all the node pairs in the graph, we can get the longest path in the original graph if we can solve the deterministic flow interdiction problem. This implies that the two flow interdiction problems on general directed graph are extremely hard to approximate within a non-trivial factor in polynomial or even quasi-polynomial time. Finally, we note that our interdiction paradigm can also be generalized to interdiction with multiple sources and destinations by adding a super source node and a super destinations nodes and connecting the super source and super destination to the original sources and destinations respectively.

[3] $DTIME(n^{\log n})$ denotes the class of problems that can be solved in quasi-polynomial time.

2.6 Chapter Summary

In this chapter, we proposed a new paradigm for network interdiction that models the interdictor's action as injecting bounded-value flows to maximally reduce the throughput of the residual network. We studied two problems under the paradigm: deterministic flow interdiction and robust flow interdiction, where the interdictor has certain or uncertain knowledge of the operation of network users, respectively. Having proved the computation complexity of the two problems, we proposed an algorithm with logarithmic approximation ratio and quasi-polynomial running time was proposed for the deterministic flow interdiction. We further developed an approximation framework that integrates the algorithm and forms a quasi-polynomial time procedure that approximates the robust flow interdiction within a poly-logarithmic factor. Finally, we evaluated the performance of the proposed algorithms through simulations.

2.7 Chapter Appendix

2.7.1 Proof of Proposition 1

Let w be an interdiction strategy. If it is a distribution on single-path flows, then the proposition follows. Otherwise, there exists an $\mathbf{f}^*$ with $w(\mathbf{f}^*) > 0$ that is not a single-path flow. By the flow decomposition theorem [41] and that the network is acyclic, we can decompose $\mathbf{f}^*$ into $\mathbf{f}^* = \sum_i \mathbf{q}_i$, where $\mathbf{q}_1, \ldots, \mathbf{q}_r$ are single-path flows from s to t. We further define $\mathbf{q}_i^* = \frac{\gamma}{val(\mathbf{q}_i)}\mathbf{q}_i$ as a scaled version of $\mathbf{q}_i$ with value γ, for $i \in \{1, \ldots, r\}$. Note that $\sum_i^r val(\mathbf{q}_i) = \gamma$, and since $\gamma \leq \min_e C(e)$, $\mathbf{q}_1^*, \ldots, \mathbf{q}_r^*$ are all valid single-path flows in $\mathcal{F}_\gamma$. In the following, we show that we can redistribute the probability that w lays on $\mathbf{f}^*$ to all its component single-path flows by decreasing $w(\mathbf{f}^*)$ to zero and adding $\frac{val(\mathbf{q}_i)}{\gamma}w(\mathbf{f}^*)$ to each $w(\mathbf{q}_i)$, with the resulting interdiction strategy w' satisfying $\Lambda(w', P) \geq \Lambda(w, P)$. Repeating the process for all non-single-path flows $\mathbf{f}^*$ with $w(\mathbf{f}^*) > 0$, we prove the proposition.

For any P, we write the linear program (2.1) with respect to $\mathbf{f}^*$ and P in vector form and construct its dual as follows:

$$\begin{array}{rl} \text{maximize} & \mathbf{1}^\top \tilde{\boldsymbol{\lambda}} \\ \textbf{s.t.} & \mathbf{A}\tilde{\boldsymbol{\lambda}} \leq \tilde{C}_{\mathbf{f}^*} \\ & \mathbf{0} \leq \tilde{\boldsymbol{\lambda}} \leq \boldsymbol{\lambda} \end{array} \qquad\qquad \begin{array}{rl} \text{minimize} & \tilde{C}_{\mathbf{f}^*}^\top \mathbf{g}_0 + \boldsymbol{\lambda}^\top \mathbf{g}_1 \\ \textbf{s.t.} & \mathbf{A}^\top \mathbf{g}_0 + \mathbf{I}\mathbf{g}_1 \geq \mathbf{1} \\ & \mathbf{g}_0, \mathbf{g}_1 \geq \mathbf{0} \end{array}$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k)^\top$ is the vector of the initial flow values, $\tilde{C}_{\mathbf{f}^*} = C - \mathbf{f}^*$, $\mathbf{A}$ is the matrix representation of constraints (2.2), $\mathbf{I}, \mathbf{1}, \mathbf{0}$ are the identity matrix, the all-1 vector and the all-0 vector, and $\mathbf{g}_0, \mathbf{g}_1$ are the dual variables. By the strong duality theorem [42], the optimal value $T(\mathbf{f}^*, P)$ of the primal problem is equal to $\tilde{C}_{\mathbf{f}^*}^\top \mathbf{g}_0^* + \boldsymbol{\lambda}^\top \mathbf{g}_1^*$, where $\mathbf{g}_0^*, \mathbf{g}_1^*$ is an optimal basic feasible solution to the dual problem. Furthermore, consider the linear program (2.1) with respect to each $\mathbf{q}_i^*$ and its dual. Note that the dual has the same feasible region as that associated with $\mathbf{f}^*$. Therefore, $\mathbf{g}_0^*, \mathbf{g}_1^*$ is still a basic feasible solution. Now, invoking weak duality, we have

$$\forall i, \quad T(\mathbf{q}_i^*, P) \leq (C - \mathbf{q}_i^*)^\top \mathbf{g}_0^* + \boldsymbol{\lambda}^\top \mathbf{g}_1^*.$$

It follows that

$$\begin{aligned} &\sum_i \frac{val(\mathbf{q}_i) w(\mathbf{f}^*)}{\gamma} T(\mathbf{q}_i^*, P) \\ &\leq \sum_i \frac{val(\mathbf{q}_i) w(\mathbf{f}^*)}{\gamma} \left((C - \mathbf{q}_i^*)^\top \mathbf{g}_0^* + \boldsymbol{\lambda}^\top \mathbf{g}_1^* \right) \\ &= w(\mathbf{f}^*)[(C - \mathbf{f}^*)^\top \mathbf{g}_0^* + \boldsymbol{\lambda}^\top \mathbf{g}_1^*] = w(\mathbf{f}^*) T(\mathbf{f}^*, P). \end{aligned}$$

Hence, we have $\Lambda(w', P) \geq \Lambda(w, P)$, and the proposition follows.

2.8 Proof of Proposition 3

The proof is done by reduction from the 3-satisfiability problem, which is a classical NP-Complete problem [45]. *3-satisfiability: Given a set of boolean variables $x_i, 1 \leq i \leq n$ and a formula $C_1 \vee C_2 \vee \ldots \vee C_k$ with each clause C_j being a disjunction ($\wedge$) of at most three literals x_i or $\overline{x_i}$, the 3-satisfiability asks whether there is a satisfying assignment, i.e., an assignment of the variables that makes the formula true.*

Given an instance of 3-satisfiability, the corresponding instance of the deterministic flow interdiction is constructed as follows. To begin with, without loss of generality, we assume that there is no clause that contains both x_i and $\overline{x_i}$ for some i, as such clause can be satisfied by all the assignments. To create the network, we first add a path p_j with $3n$ edges for each clause C_j. The paths are node-disjoint. Then for each variable x_j, we create a variable gadget with three nodes u_i, v_{i0}, v_{i1} and two edges $(u_i, v_{i0}), (u_i, v_{i1})$. Nodes v_{i0}, v_{i1} correspond to $\overline{x_i}$, x_i, respectively. We next connect the variable gadgets and the paths for the clauses. For each v_{i_0} (v_{i_1}), let $C_{i1}, \ldots, C_{ir}$ be the set of clauses that contains literal $\overline{x_i}(x_i)$. Let $e_1, \ldots, e_r$ be the $3i$-th ($3i$ + 1-th) edges on $p_{i1}, \ldots, p_{ir}$. We add edges to the network to sequentially connect $v_{i_0}, e_1, \ldots, e_r, u_{i+1}$ and refer to the resulting path from v_{i_0} to u_{i+1} as v_{i_0}-u_{i+1} segment. For $i = n$, we further add a node to serve as u_{n+1}. We designate $s = u_0$ and $t = u_{n+1}$ as the source and the destination of the interdictor. The set of user paths is $P = \{p_1, \ldots, p_k\}$ and the initial flow values $f_1 = \ldots = f_k = 1$. The capacities of all the edges, and the budget of the interdictor are set to 1. Now we have completed the construction of the corresponding instance of the deterministic flow interdiciton. Note that the constructed network is a DAG and the whole reduction process can be done in polynomial time. See Figure 2-4 for an illustration of the reduction process.

We proceed to show that there exists a single-path flow $\mathbf{f}$ with $\Lambda(\mathbf{f}, P) = k$ if and only if there is a satisfying assignment for the original 3-satisfiability instance. First, if there exists a satisfying assignment with $x_i = a_i \in \{0, 1\}$, we claim that the single-path flow $\mathbf{f}$ that corresponds to the s-t path consisting of (u_i, v_{ia_i}) and v_{ia_i}-u_{i+1} segment for all i has throughput reduction k. Since in the satisfying assignment,

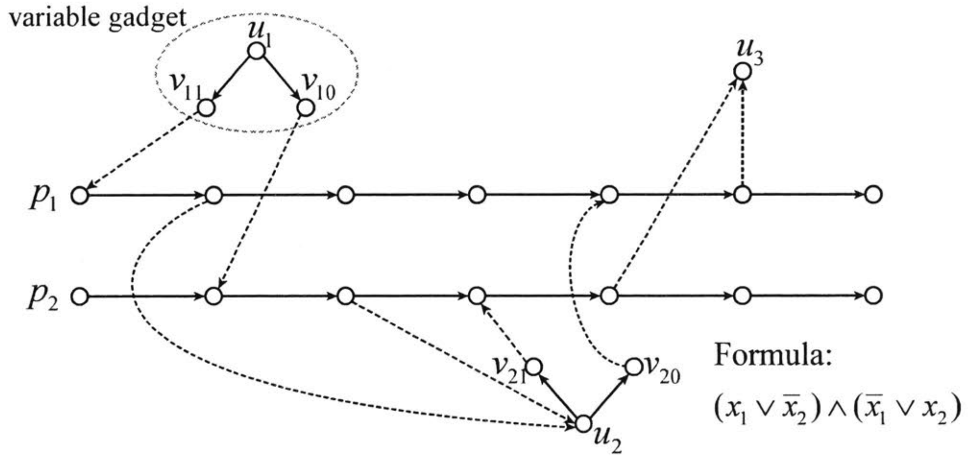


Figure 2-4: An illustration of the reduction process for the 3-satisfiability instance with formula $(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2)$.

each clause is set true by some literal, we have that each user path contains an edge with zero residual capacity after interdicted by $\mathbf{f}$. It follows that $T(\mathbf{f}, P) = 0$ and $\Lambda(\mathbf{f}, P) = k$. Second, if there exists a single-path flow $\mathbf{f}$ with $\Lambda(\mathbf{f}, P) = k$, then the path $E_{\mathbf{f}}$ that $\mathbf{f}$ corresponds to must intersect with all user paths. We next show that $E_{\mathbf{f}}$ can be converted to an s-t path $E_{\mathbf{f}'}$ consisting only of (u_i, v_{ia_i})'s and v_{ia_i}-u_{i+1} segments and also intersect with all user paths. Indeed, $E_{\mathbf{f}'}$ can be constructed by taking all the (u_i, v_{ia_i})'s and v_{ia_i}-u_{i+1} segments that $E_{\mathbf{f}}$ intersects. Note that since there is no clause that contains both x_i and $\overline{x_i}$ for some i, the assignment $\forall i,\ x_i = a_i$ induced by $E_{\mathbf{f}'}$ is a valid assignment. Since $E_{\mathbf{f}'}$ intersects with all user paths, the assignment satisfies all the clauses, and thus makes the formula true. Hence, we justify the validity of the reduction. Combining with corollary 1, we have that the deterministic flow interdiction problem is NP-hard.

2.8.1 Generalization of the Recursive Greedy Algorithm

In this section, we describe a generalized version of the Recursive Greedy algorithm that uses more than one anchors. Let $a > 1$ be some integers. The details of the algorithm is presented in **Algorithm 3**. At step 8, instead of going over all $v \in V$, the generalized algorithm goes through all $a - 1$ combinations of nodes in V and uses them as anchors. The analysis of the algorithm is given in Theorem 4

Algorithm 3 The Generalized Recursive Greedy Algorithm

```
Input: Network graph $G(V,E)$, user paths $P = \{p_1, \dots, p_2\}$ with initial flow values
    $\{f_1, \dots, f_k\}$, Interdictor's source $s$, destination $t$ and budget $\gamma$
Output: The optimal $s$-$t$ path $E_{\mathrm{f}}$
 1: Run: $RG(s, t, \emptyset, I)$
    The Recursive Function $RG(u_1, u_2, X, i)$:
 2: $E_{\mathrm{f}} :=$ shortest $s$-$t$ path.
 3: if $E_{\mathrm{f}}$ does not exist then
 4:     return Infeasible
 5: if $i = 0$ then
 6:     return $E_{\mathrm{f}}$
 7: $r := \Lambda_X(E_{\mathrm{f}}, P)$.
 8: for $v_1, v_2, \dots, v_{a-1} \in V$ do
 9:     $E_{\mathrm{f_1}} := RG(u_1, v_1, X, i-1), E_{\mathrm{f_2}} := RG(v_1, v_2, X \cup E_{\mathrm{f_1}}, i-1), \dots, E_{\mathrm{f_a}} :=$
        $RG(v_{a-1}, u_2, X \cup E_{\mathrm{f_1}} \cup \dots \cup E_{\mathrm{f_{a-1}}}, i-1)$.
10:     if $\Lambda_X(E_{\mathrm{f_1}} \cup \dots \cup E_{\mathrm{f_a}}, P) > r$ then
11:         $r := \Lambda_X(E_{\mathrm{f_1}} \cup \dots \cup E_{\mathrm{f_a}}, P)$, $E_{\mathrm{f}} := E_{\mathrm{f_1}} \cup \dots \cup E_{\mathrm{f_a}}$.
12: return $E_{\mathrm{f}}$
```

Theorem 4. *If $I \geq \lceil \log_a d \rceil$, the Generalized Recursive Greedy algorithm returns an s-t path E_{f} with $\Lambda(E_{\mathrm{f}}, P) \geq \frac{1}{\lceil \log_a d \rceil + 1} \Lambda(E_{\mathrm{f^*}}, P)$, where d is the length of $E_{\mathrm{f^*}}$.*

Proof. We prove a more general claim, that for all $u_1, u_2 \in V$, $X \subseteq E$, if $I \geq \lceil \log d \rceil$, the procedure $RG(u_1, u_2, X, I)$ returns an u_1-u_2 path E_{f} with $\Lambda_X(E_{\mathrm{f}}, P) \geq \frac{1}{\lceil \log d \rceil + 1} \Lambda_X(E_{\mathrm{f^*}}, P)$, where $E_{\mathrm{f^*}}$ is the u_1-u_2 path that maximized $\Lambda(\cdot, P)$ and d is the length of $E_{\mathrm{f^*}}$. The theorem follows from the claim by setting $u_1 = s$, $u_2 = t$ and $X = \emptyset$.

Let the nodes on the path $E_{\mathrm{f^*}}$ be $\{u_1 = v_0, \dots, v_d = u_2\}$. The proof is done by induction on d. First, for the base step, when $d = 1$, it means that there exists an edge between u_1 and u_2, which must be the shortest u_1-u_2 path. Obviously the procedure checks this path, and the claim follows. Next, suppose the claim holds for $d \leq l \in \mathbb{N}$. When $d = l + 1$, let $v_1^* = v_{\lceil \frac{d}{a} \rceil}, v_2^* = v_{\lceil \frac{2d}{a} \rceil}, \dots, v_{a-1}^* = v_{\lceil \frac{(a-1)d}{a} \rceil}$. Let $E_{\mathrm{f_1^*}}, \dots, E_{\mathrm{f_a^*}}$ be the subpaths of $E_{\mathrm{f^*}}$ from s to $v_1^*, \dots, v_{a-1}^*$ to t. When RG examines $\{v_1^*, \dots, v_{a-1}^*\}$ at step 8, it invokes a sub-procedures denoted as $RG(u_1, v_1^*, X^{(0)}, I - 1), \dots, RG(v_{a-1}^*, u_2, X^{(a-1)}, I - 1)$. In the above notations, we use E_{f_j} to denote the sub-path returned by the jth subprocedure, $X^{(j)}$ to denote $X \cup E_{\mathbf{f}_1} \cup \ldots \cup E_{\mathbf{f}_j}$ for $j \in \{1, \ldots, a\}$ and $X^{(0)} = X$. Let $E'_{\mathbf{f}} = E_{\mathbf{f}_1} \cup \ldots \cup E_{\mathbf{f}_a}$. Our goal is to show that

$$\Lambda_X(E'_{\mathbf{f}}, P) \geq \frac{1}{\lceil \log_a d \rceil + 1} \Lambda_X(E_{\mathbf{f}^*}, P), \tag{2.16}$$

which proves the induction step since the path $E_{\mathbf{f}}$ that $RG(s, t, X, I)$ returns must satisfy $\Lambda_X(E_{\mathbf{f}}, P) \geq \Lambda_X(E'_{\mathbf{f}}, P)$.

Since $I \geq \lceil \log_a d \rceil$, we have $I - 1 \geq \lceil \log_a d \rceil - 1 = \lceil \log_a d/a \rceil$. By the induction hypothesis,

$$\begin{aligned}
\Lambda_{X^{(0)}}(E_{\mathbf{f}_1}, P) &\geq \frac{1}{\lceil \log_a d \rceil} \Lambda_{X^{(0)}}(E_{\mathbf{f}_1^*}, P), \\
\Lambda_{X^{(1)}}(E_{\mathbf{f}_2}, P) &\geq \frac{1}{\lceil \log_a d \rceil} \Lambda_{X^{(1)}}(E_{\mathbf{f}_2^*}, P), \\
&\ldots, \\
\Lambda_{X^{(a-1)}}(E_{\mathbf{f}_a}, P) &\geq \frac{1}{\lceil \log_a d \rceil} \Lambda_{X^{(a-1)}}(E_{\mathbf{f}_a^*}, P).
\end{aligned}$$

For $j \in \{0, \ldots, a-1\}$, by the submodularity of Λ (Lemma 1), we have $\Lambda_{X^{(j)}}(E_{\mathbf{f}_a^*}, P) \geq \Lambda_{X^{(a)}}(E_{\mathbf{f}_a^*}, P)$. Using this, we sum all the inequalities above and get

$$\begin{aligned}
\Lambda_X(E'_{\mathbf{f}}, P) &= \sum_{j=0}^{a-1} \Lambda_{X^{(j)}}(E_{\mathbf{f}_{j+1}}, P) \qquad (2.17)\\
&\geq \frac{1}{\lceil \log_a d \rceil} \sum_{j=0}^{a-1} \Lambda_{X^{(j)}}(E_{\mathbf{f}_{j+1}^*}, P) \qquad (2.18)\\
&\geq \frac{1}{\lceil \log_a d \rceil} \sum_{j=0}^{a-1} \Lambda_{X^{(a)}}(E_{\mathbf{f}_{j+1}^*}, P). \qquad (2.19)
\end{aligned}$$

Again, by Lemma 1, we have for $j \in \{0, \ldots, a-1\}$,

$$\Lambda_{X^{(a)}}(E_{\mathbf{f}_{j+1}^*}, P) \geq \Lambda_{X^{(a)} \cup \left(\bigcup_{i=1}^{j} E_{\mathbf{f}_i^*}\right)}(E_{\mathbf{f}_{j+1}^*}, P)$$

It follows that

$$
\begin{aligned}
\Lambda_X(E'_{\mathbf{f}}, P) &\geq \frac{1}{\lceil \log_a d \rceil} \sum_{j=0}^{a-1} \Lambda_{X^{(a)} \cup \left(\bigcup_{i=1}^{j} E_{\mathbf{f}_i^*}\right)}(E_{\mathbf{f}_{j+1}^*}, P) && (2.20)\\
&= \frac{1}{\lceil \log_a d \rceil} \Lambda_{X^{(a)}}(E_{\mathbf{f}_1^*} \cup \ldots \cup E_{\mathbf{f}_a^*}, P) && (2.21)\\
&= \frac{1}{\lceil \log_a d \rceil} \left[\Lambda \left(X \cup \left(\bigcup_{j=1}^{a} E_{\mathbf{f_j}} \right) \cup \left(\bigcup_{j=1}^{a} E_{\mathbf{f_j^*}} \right), P \right) \right.\\
&\quad \left. - \Lambda \left(X \cup \left(\bigcup_{j=1}^{a} E_{\mathbf{f_j}} \right), P \right) \right] && (2.22)\\
&= \frac{1}{\lceil \log_a d \rceil} \left[\Lambda_X \left(\left(\bigcup_{j=1}^{a} E_{\mathbf{f_j}} \right) \cup \left(\bigcup_{j=1}^{a} E_{\mathbf{f_j^*}} \right), P \right) \right.\\
&\quad \left. - \Lambda_X \left(\bigcup_{j=1}^{a} E_{\mathbf{f_j}}, P \right) \right] && (2.23)\\
&\geq \frac{1}{\lceil \log_a d \rceil} \left[\Lambda_X \left(\bigcup_{j=1}^{a} E_{\mathbf{f_j^*}}, P \right) - \Lambda_X \left(\bigcup_{j=1}^{a} E_{\mathbf{f_j}}, P \right) \right] && (2.24)\\
&= \frac{1}{\lceil \log_a d \rceil} \left[\Lambda_X \left(E_{\mathbf{f}^*}, P \right) - \Lambda_X \left(E'_{\mathbf{f}}, P \right) \right], && (2.25)
\end{aligned}
$$

where inequality (2.24) follows from the monotonicity of Λ and equalities (2.21), (2.22) and (2.23) follow from the definition of Λ_X. From (2.25), we obtain (2.16), which concludes the proof. $\square$

Time Complexity: As we invoke at most an^a sub-procedures at each level of recursion and the computation of Λ takes $O(m)$ time, the time complexity of the Generalized Recursive Greedy algorithm is $O((an)^{(a-1)I}m)$. Again, taking $I = \log_a n$, we get an $1/(\lceil \log_a n \rceil + 1)$-approximation with a time complexity of $O((an)^{(a-1)\log_a n}m)$.

2.9 Proof of Lemmas 2 and 3

This section is devoted to the proof of Lemmas 2 and 3. We define $\prod_{e \in p_i, e \in E_2, \tilde{C}_A(e) \leq \sum_{p_j \ni e} \lambda_j} \frac{\tilde{C}_A(e)}{\sum_{p_j \ni e} \lambda_j}$ as $\Delta_i(A, P)$.

2.9.1 Lemma 2

Recall the definition of submodularity and monotonicity in Lemma 1. First, we can easily see from Phase I and Phase II that for all i, $\tilde{\lambda}_{iA}^{(2)}$ is monotonically non-increasing with respect to A. It follows that $\bar{\Lambda}(\cdot, P)$ is monotone.

Next, we prove the submodularity of $\bar{\Lambda}$. Consider two sets $A \subseteq B \subseteq E$ and an edge $e \in E, e \notin B$. Our goal is to show that $\bar{\Lambda}(A \cup \{e\}, P) - \bar{\Lambda}(A, P) \geq \bar{\Lambda}(B \cup \{e\}, P) - \bar{\Lambda}(B, P)$. We divide the proof into three cases.

Case I: If $e \notin E_0$, then $\bar{\Lambda}(A \cup \{e\}, P) - \bar{\Lambda}(A, P) = \bar{\Lambda}(B \cup \{e\}, P) - \bar{\Lambda}(B, P) = 0$.

Case II: If $e \in E_1$, then suppose $e \in p_i$ for some i. Note that since $A \subseteq B$, $\tilde{\lambda}_{iA}^{(1)} \geq \tilde{\lambda}_{iB}^{(1)}$. We further divide this case into three subcases. (i). If $C(e) - \gamma \geq \tilde{\lambda}_{iA}^{(1)}$, then we have $\tilde{\lambda}_{iA}^{(1)} = \tilde{\lambda}_{iA\cup\{e\}}^{(1)}$ and $\tilde{\lambda}_{iB}^{(1)} = \tilde{\lambda}_{iB\cup\{e\}}^{(1)}$. Hence,

$$\bar{\Lambda}(A \cup \{e\}, P) - \bar{\Lambda}(A, P) = \bar{\Lambda}(B \cup \{e\}, P) - \bar{\Lambda}(B, P) = 0.$$

(ii). If $\tilde{\lambda}_{iA}^{(1)} > C(e) - \gamma \geq \tilde{\lambda}_{iB}^{(1)}$, then $\tilde{\lambda}_{iA}^{(1)} > \tilde{\lambda}_{iA\cup\{e\}}^{(1)}$ and $\tilde{\lambda}_{iB}^{(1)} = \tilde{\lambda}_{iB\cup\{e\}}^{(1)}$. Hence,

$$\bar{\Lambda}(A \cup \{e\}, P) - \bar{\Lambda}(A, P) > 0,$$
$$\bar{\Lambda}(B \cup \{e\}, P) - \bar{\Lambda}(B, P) = 0.$$

(iii). If $C(e) - \gamma < \tilde{\lambda}_{iB}^{(1)}$, we have $\tilde{\lambda}_{iA\cup\{e\}}^{(1)} = \tilde{\lambda}_{iB\cup\{e\}}^{(1)} = C(e) - \gamma$. It follows that

$$\tilde{\lambda}_{iA\cup\{e\}}^{(2)} = (C(e) - \gamma) \cdot \Delta_i(A \cup \{e\}, P),$$
$$\tilde{\lambda}_{iB\cup\{e\}}^{(2)} = (C(e) - \gamma) \cdot \Delta_i(B \cup \{e\}, P).$$

As $A \cup \{e\} \subseteq B \cup \{e\}$, we have $\Delta_i(A \cup \{e\}, P) \geq \Delta_i(B \cup \{e\}, P)$. It follows that,

$$\begin{aligned}\bar{\Lambda}(A \cup \{e\}, P) - \bar{\Lambda}(A, P) &= (\tilde{\lambda}^{(1)}_{iA} - \tilde{\lambda}^{(1)}_{iA\cup\{e\}})\Delta_i(A \cup \{e\}, P) \\ &\geq (\tilde{\lambda}^{(1)}_{iB} - \tilde{\lambda}^{(1)}_{iB\cup\{e\}})\Delta_i(B \cup \{e\}, P) \\ &= \bar{\Lambda}(B \cup \{e\}, P) - \bar{\Lambda}(B, P),\end{aligned}$$

where the two equalities follow from the fact that $\Delta_i(A \cup \{e\}, P) = \Delta_i(A, P), \Delta_i(B \cup \{e\}, P) = \Delta_i(B, P)$ since $e \notin E_2$.

Case III: If $e \in E_2$, then when $C(e) - \gamma \geq \sum_{p_j \ni e} \lambda_j$, we still have

$$\bar{\Lambda}(A \cup \{e\}, P) - \bar{\Lambda}(A, P) = \bar{\Lambda}(B \cup \{e\}, P) - \bar{\Lambda}(B, P) = 0.$$

When $C(e) - \gamma < \sum_{p_j \ni e} \lambda_j$, first, we observe that since $e \notin E_1$, $\tilde{\lambda}^{(1)}_{iA} = \tilde{\lambda}^{(1)}_{iA\cup\{e\}}$ and $\tilde{\lambda}^{(1)}_{iB} = \tilde{\lambda}^{(1)}_{iB\cup\{e\}}$. Also, as $A \subseteq B$, we have $\tilde{\lambda}^{(1)}_{iA} \geq \tilde{\lambda}^{(1)}_{iB}$ and $\Delta_i(A, P) \geq \Delta_i(B, P)$ for all i. Combining these, we obtain

$$\begin{aligned}&\bar{\Lambda}(A \cup \{e\}, P) - \bar{\Lambda}(A, P) \\ &= \sum_{i: e \in p_i} \tilde{\lambda}^{(1)}_{iA} \cdot \Delta_i(A, P) \cdot \left(1 - \frac{C(e) - \gamma}{\sum_{p_j \ni e} \lambda_j}\right) \\ &\geq \sum_{i: e \in p_i} \tilde{\lambda}^{(1)}_{iB} \cdot \Delta_i(B, P) \cdot \left(1 - \frac{C(e) - \gamma}{\sum_{p_j \in e} \lambda_j}\right) \\ &= \bar{\Lambda}(B \cup \{e\}, P) - \bar{\Lambda}(B, P),\end{aligned}$$

Therefore, in all cases, we have $\bar{\Lambda}(A \cup \{e\}, P) - \bar{\Lambda}(A, P) \geq \bar{\Lambda}(B \cup \{e\}, P) - \bar{\Lambda}(B, P)$. Hence, $\bar{\Lambda}(\cdot, P)$ is submodular.

Lemma 3

In the definition of $\bar{\Lambda}$, we have reasoned that $\Lambda(A, P) \leq \bar{\Lambda}(A, P)$. What is left is to show that $\bar{\Lambda}(A, P) \leq (b + 1) \cdot \Lambda(A, P)$. Let $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_k$ be an optimal solution to the maximization problem (2.1) associated with A.

First, for Phase I, observe that for all i, $\tilde{\lambda}_i \leq \min\{\lambda_i, \min_{e\in p_i, e\in E_1}\{\tilde{C}_A(e)\}\}$. Therefore, we have $\sum_i \tilde{\lambda}^{(1)}_{iA} \geq T(A,P)$. It follows that

$$\Lambda(A,P) \geq \sum_i (\lambda_i - \tilde{\lambda}^{(1)}_{iA}). \tag{2.26}$$

Next, for phase II,

$$\begin{aligned}
&\sum_i \left(\tilde{\lambda}^{(1)}_{iA} - \tilde{\lambda}^{(2)}_{iA}\right) = \sum_i \tilde{\lambda}^{(1)}_{iA} \left(1 - \Delta_i(A,P)\right) \\
\leq &\sum_i \lambda_i \left(1 - \Delta_i(A,P)\right) \\
\leq &\sum_i \lambda_i \cdot \sum_{e\in p_i, e\in E_2, \tilde{C}_A(e)\leq \sum_{p_j\ni e}\lambda_j} \left(1 - \frac{\tilde{C}_A(e)}{\sum_{p_j\ni e}\lambda_j}\right) &(2.27)\\
= &\sum_{e\in E_2, \tilde{C}_A(e)\leq \sum_{p_j\ni e}\lambda_j} \left(\sum_{p_j\ni e}\lambda_j - \tilde{C}_A(e)\right), &(2.28)
\end{aligned}$$

where inequality (2.27) follows from the fact that $1 - \prod_j a_j \leq \sum(1-a_j)$ for $0 \leq a_1 \leq \ldots \leq a_j \leq 1$ and equality (2.28) comes from rearranging the terms. Since $\sum_{p_j\ni e}\tilde{\lambda}_j \leq \tilde{C}_A(e)$ for all e, we have

$$\begin{aligned}
&\sum_{e\in E_2, \tilde{C}_A(e)\leq \sum_{p_j\ni e}\lambda_j} \left(\sum_{p_j\ni e}\lambda_j - \tilde{C}_A(e)\right) \\
\leq &\sum_{e\in E_2, \tilde{C}_A(e)\leq \sum_{e\in p_j}\lambda_j} \left(\sum_{p_j\ni e}\lambda_j - \sum_{p_j\ni e}\tilde{\lambda}_j\right) \\
\leq &\, b \cdot \sum_i \left(\lambda_i - \tilde{\lambda}_i\right) = b\cdot\Lambda(A,P). &(2.29)
\end{aligned}$$

Therefore, combining (2.26) and (2.29), we have $\bar{\Lambda}(A,P) = \sum_i \left(\lambda_i - \tilde{\lambda}^{(1)}_{iA} + \tilde{\lambda}^{(1)}_{iA} - \tilde{\lambda}^{(2)}_{iA}\right) \leq (b+1)\Lambda(A,P)$.

2.9.2 Justification of Integrality Assumption of Λ

In this section, we show that not much generality is lost if we consider the throughput reduction function Λ to take value in integers that are bounded by some polynomial of n. Specifically, we show the following proposition.

Proposition 5. *For any instance $\mathcal{I}$ of the robust flow interdiction problem with throughput reduction function Λ and some fixed $\epsilon > 0$, we can construct another instance $\mathcal{I}'$ in whose throughput reduction function$'$ take integral values that are bounded by some polynomial of n for all $w, P \in \mathcal{U}$. Furthermore, if we apply* **Algorithm 2** *to $\mathcal{I}'$, it returns a strategy w that satisfies*

$$\begin{aligned}&\min_{P\in\mathcal{U}} \Lambda(w, P)\\ &\geq (1-\epsilon)^2 \left(\frac{N_0}{(N_0+1)(b+1)\log\frac{M}{\epsilon}\cdot(\lceil \log d\rceil + 1)} \right) \min_{P\in\mathcal{U}} \Lambda(w^*, P)\end{aligned}$$

for the original instance $\mathcal{I}$.

Proof. First, we assume that without loss of generality, if $\Lambda(w, P) > 0$ for some w, P, then $\Lambda(w, P) > 1$, as we can always scale up all the parameters if the condition is not satisfied. With this condition, we define $\Lambda'(\mathbf{f}, P) = \lfloor N_1 \Lambda(\mathbf{f}, P) \rfloor$, where N is some integer bounded by some polynomial of n and satisfies $N_1 = \lceil \frac{2(b+1)\log(N_1 M)(\lfloor \log d \rfloor + 1)}{\epsilon} \rceil$. Keeping all other parameters unchanged and substituting Λ with Λ', we get the new instance $\mathcal{I}'$. Note that $\Lambda'(w, P)$ satisfies the condition in the statement of the proposition. Thus, we can apply the proposed framework **Algorithm 2** to $\mathcal{I}'$ (with $M' = NM$). Note that for the Extended Recursive Greedy Algorithm in the framework, the approximate function $\bar{\Lambda}$ we use is calculated with respect to Λ in the original instance $\mathcal{I}$.

To establish the performance guarantee of such procedure, we first analyze the quality of the greedy iterations computed by the Extended Greedy algorithm. At some iteration, let $\mathbf{f}$ be the single-path flow that corresponds to the path returned by the algorithm. Note that as Lemma 3 holds for all $P \in \mathcal{U}$, we have by Theorem 2 that

$$\begin{aligned}\Lambda(\mathbf{f},P) &\geq \frac{1}{(b+1)\cdot(\lceil\log d\rceil+1)}\Lambda(\mathbf{f}^*,P)\\ &\geq \frac{1}{(b+1)\cdot(\lceil\log d\rceil+1)}\Lambda(\mathbf{f}',P),\end{aligned}$$

where $\mathbf{f}^*$ is the flow returned by the exact greedy scheme with respect to Λ and $\mathbf{f}'$ is the flow returned by that with respect to $'$. It follows that

$$\begin{aligned}\sum_{P\in\mathcal{U}}\Lambda'(\mathbf{f},P) &= \sum_{P\in\mathcal{U}}\lfloor N_1\Lambda(\mathbf{f},P)\rfloor \\ &\geq \sum_{P\in\mathcal{U}}\lfloor\frac{N_1}{(b+1)\cdot(\lceil\log d\rceil+1)}\Lambda(\mathbf{f}',P)\rfloor \\ &\geq \sum_{P\in\mathcal{U}}\left(\frac{N_1}{(b+1)\cdot(\lceil\log d\rceil+1)}\Lambda(\mathbf{f}',P)-1\right) \\ &\geq \sum_{P\in\mathcal{U}}(1-\epsilon)\frac{N_1\Lambda(\mathbf{f}',P)}{(b+1)\cdot(\lceil\log d\rceil+1)} \\ &= \frac{1-\epsilon}{(b+1)\cdot(\lceil\log d\rceil+1)}\sum_{P\in\mathcal{U}}\Lambda'(\mathbf{f}',P).\end{aligned} \tag{2.30}$$

Therefore, the quality of the obtained approximate greedy solutions enjoys almost the same guarantee with respect to $'$. Then, invoking Theorem 3, denoting the optimal N_0-bounded strategy for $\mathcal{I}'$ as w', we have that

$$\begin{aligned}&\min_{P\in\mathcal{U}}\Lambda(w,P) \geq \frac{1}{N_1}\min_{P\in\mathcal{U}}\Lambda'(w,P)\\ &\geq \left(\frac{N_0\,(1-\epsilon)}{(N_0+1)(b+1)\log M'\cdot(\lceil\log d\rceil+1)}\right)\min_{P\in\mathcal{U}}\frac{\Lambda'(w',P)}{N_1}\\ &\geq \left(\frac{N_0\,(1-\epsilon)}{(N_0+1)(b+1)\log\frac{M}{\epsilon}\cdot(\lceil\log d\rceil+1)}\right)\min_{P\in\mathcal{U}}\frac{\Lambda'(w^*,P)}{N_1}\\ &\geq \left(\frac{N_0\,(1-\epsilon)}{(N_0+1)(b+1)\log\frac{M}{\epsilon}\cdot(\lceil\log d\rceil+1)}\right)\min_{P\in\mathcal{U}}\frac{N_1\Lambda(w^*,P)-1}{N_1}\\ &\geq \left(\frac{N_0\,(1-\epsilon)^2}{(N_0+1)(b+1)\log\frac{M}{\epsilon}\cdot(\lceil\log d\rceil+1)}\right)\min_{P\in\mathcal{U}}\Lambda(w^*,P).\end{aligned}$$

□

Note that the bound obtained in the proposition is essentially the same as that in Theorem 3.

Chapter 3

Fundamental Limit of Volume-based Network DoS Attacks

In this chapter, we shift out attention to flow-based attacks on networks with dynamic routing. This framework applies to practical scenarios such as TCP SYN Flood and DNS Flood [1, 46], where the victim network is typically server farm that employs dynamic load-balancing and scheduling schemes. This kind of attacks is often referred to as volume-based DoS attack in the research community [1]. The dynamic nature of the target system renders the interdiction paradigm in the previous chapter inapplicable. Instead, we model the system as a queueing network, which enables us to incorporate widely studied dynamic routing algorithms such as JSQ and Max-Weight.

We start our analysis with a single-hop network of general bipartite topology, where one side of the nodes consists of user and adversary traffic dispatchers and the other side consists of parallel servers with a queue at each server. User traffic arrives at each user dispatcher at a certain rate and is sent to the servers following the Join-the-Shortest-Queue (JSQ) rule. Adversary traffic of certain rate arrives at each adversary dispatchers and gets sent to the servers under some adversary injection policy. Each server serves the queue at some service rate, with the servers assumed to not be able to distinguish user and adversary traffic and employs FCFS service discipline. The success of the network DoS attack, which is also the goal of the adversary, is defined as making the user traffic in a queue grow to infinity with time, i.e., blocking a certain amount of user traffic from getting served (see Section 3.1 for rigorous definitions). Projecting into practical scenarios, the servers in our model can represent web-servers, DNS servers, etc., and the queues correspond to connection queue, memory or the bandwidth of the servers, depending on the application. The user dispatchers naturally correspond to load-balancers in various network applications, and the adversary dispatchers mirror the initiators of the attack such as botnets. Our model can thus be seen to capture many DoS attack scenarios in real life such as TCP SYN Flood and DNS Flood [46].

Under our model, we answer the previously raised questions on the fundamental limit of network DoS attack through the following main results:

1. We first gives a necessary and sufficient condition on the network topology, the user traffic rates, adversary traffic rates and servers' service rates for the feasibility of network DoS attack. As the arrival rates of adversary traffic can be naturally considered as the budget on the adversary's resource, the feasibility condition can be interpreted as an resource requirement for the adversary to launch a successful DoS attack and at the same time captures the sustainability boundary of the network.

2. We then design an optimal adversary injection policy that does not rely on the knowledge of the network statistics: user traffic rates, service rates and even the adversary's budget. It is optimal in the sense that the policy achieves the goal of the network DoS attack whenever the feasibility condition is met. The existence of such policy demonstrates that the lack of network statistics does not reduce the adversary's capability to conduct network DoS attack, and thus the feasibility condition is still valid for adversary that is statistics-oblivious. We also evaluate the performance and properties of the proposed policy through simulations.

3. Finally, we generalize our results to multi-hop network that employs the back-pressure routing policy [16] and extend the feasibility condition and the optimal adversary injection policy to the multi-hop scenario.

Table 3.1: Notations and Definitions

Notation	Definition
S, U, V	Sets of servers, user dispatchers and adversary dispatchers
s_n or n, u_l or l, v_m or m	generic server, user dispatcher and adversary dispatcher
S_{u_l}, S_{v_m}	The set of servers u_l (v_m) has connection to
$Q_n(t)$	Queue length at server n at time t
$b_n(t), \mu_n$	Offered service of server n at time t and its mean
$\lambda_l^u(t), \lambda_u^l$	User traffic arrival at u_l at time t and its mean
$\lambda_m^v(t), \lambda_v^m$	Adversary traffic arrival at v_m at time t and its mean
$b_n^u(t), b_n^v(t)$	Offered service for user (adversary) traffic of server n at time t
$a_n^u(t), a_n^v(t)$	Total user (adversary) packets routed to server n at time t
$a_{ln}^u(t), a_{mn}^v(t)$	Amount of user (adversary) packets routed from $u_l(v_m)$ to n at t
$Q_n^u(t), Q_n^v(t)$	Amount of user (adversary) packets in Q_n at time t
$\boldsymbol{Q}(t)$	Queue length vector at time t
$\boldsymbol{\mu}, \boldsymbol{\lambda}^u, \boldsymbol{\lambda}^v$	Vectors of service rates, user traffic arrival rates and adversary budget
$U_{S'}$	Set of user dispatchers that only have connections to servers in S'

A further note is that our analysis harnesses combination and extension of results from two papers by Shah and Wischik [47, 48], and a result in Markov chain theory that establishes transience of Markov chains [49], which may be of independent interests.

The rest of the chapter is organized as follows. In Section 3.1, we formally present our model and problem formulation. We then introduce the feasibility region in Section 3.2. We next summarize some key auxiliary results in Section 3.3 and introduce the optimal adversary injection policy in Section 3.4. In Section 3.5, we evaluate the injection policy through simulations. Section 3.6 is devoted to generalization to multi-hop networks. We conclude the chapter in Section 3.7.

3.1 Model and Problem Formulation

In this section, we formally present our system model for single-hop networks, which captures server farms as a major application. The model for multi-hop networks will be presented in Section 3.6. To unify the terminology, we will refer to entity that flows in the network as packet. We also assume for simplicity that packets are of the same length. Our results can be easily generalized to the case of non-uniform packet lengths. The notations that we use throughout the paper are summarized in Table 3.1.

3.1.1 Network Model

As our single-hop network model mainly mirrors server farms, we adopt terminologies therein, and will use single-hop network and server farm interchangeably. Consider a single-hop network with a set of parallel servers (sinks) and a set of traffic dispatchers (sources). The dispatchers are divided into two disjoint subsets: user traffic dispatchers that route user traffic to servers, and adversary traffic dispatchers controlled by the adversary that sends adversary traffic to servers to block the user traffic. We use $S = \{s_1, \dots, s_N\}$ to denote the set of servers, $U = \{u_1, \dots, u_L\}$ to denote the set of user traffic dispatchers and $V = \{v_1, \dots, v_M\}$ to denote the set of adversary traffic dispatchers. For notational convenience, some time we also use the indices n, l, m to denote some server, user dispatcher and adversary dispatcher. We define $S_{u_l} \subseteq S$ as the set of servers that user dispatcher u_l has connection to, and $S_{v_m} \subseteq S$ as the set of servers that adversary dispatcher v_m as connection to. Each dispatcher can only route packets to the servers that it has connection to.

3.1.2 Queueing Dynamics

We consider a discrete-time system. Each server has a queue that buffers the packets, with $Q_n(t)$ representing the length of the queue of server s_n at time t. Additionally, we define $\mathbf{Q}(t) = (Q_1(t), \dots, Q_N(t))$ as the queue-length vector at time t. The offered service of server n at time t is denoted by $b_n(t)$. The servers do not distinguish user and adversary traffic and employ a First Come First Serve (FCFS) service discipline[1]. At each time slot, $\lambda_l^u(t)$ packets arrive at user dispatcher u_l, and u_l routes the packets to the servers following the "Join-the-Shortest-Queue" (JSQ) policy, that is, at each time slot, each user dispatcher u_l routes all its incoming packets to the server s with the minimum queue length among the ones that it has connection to

[1]This is not a technical choice. Our results hold under all common service disciplines except priority based service with user traffic having the priority

($s \in \arg\min_{s_n \in S} Q_n(t)$); $\lambda_m^v(t)$ packets arrive at adversary dispatcher v_m, and v_m routes the packets to servers according to some adversarial injection policy [2]. We assume that $b_n(t)$'s, $\lambda_l^u(t)$'s and $\lambda_m^v(t)$'s are independent integer-valued random variables, and are i.i.d across time slots with $\mathbb{E}[b_n(t)] = \mu_n, \mathbb{E}[\lambda_l^u(t)] = \lambda_l^u, \mathbb{E}[\lambda_m^v(t)] = \lambda_m^v$. We assume that the random variables are bounded, i.e., there exists $C > 0$ such that $|b_n(t)|, |\lambda_l^u(t)|, |\lambda_m^v(t)| \leq C$. For a finer description of system dynamics, we define $Q_n^u(t)$ and $Q_n^v(t)$ as the number of user packets and adversary packets in Q_n at t, respectively. At each time slot t, we decompose the offered service $b_n(t)$ into that offered to user traffic $b_n^u(t)$ and that offered to adversary traffic $b_n^v(t)$ with $b_n^u(t)+b_n^v(t) = b_n(t)$. $b_n^u(t)$ and $b_n^v(t)$ generally depend on the queue composition. We further define $a_n^u(t)$ as the sum of user traffic arrivals to server n and $a_n^v(t)$ as the counterpart of adversary traffic. we also write $a_{ln}^v(t)$ ($a_{mn}^u(t)$) as the amount traffic that user dispatcher u_l (adversary dispatcher v_m) sends to n at time t. We impose the following ordering on system dynamics for ease of presentation: at each time slot, first, user dispatchers route their incoming packets to the servers following JSQ; second, adversary dispatchers route adversary packets to the servers following some adversarial injection policy; finally, servers serve the packets in the queues. Based on the system dynamics, we summarize the queue length evolution as follows:

$$\begin{aligned}
Q_n^u(t+1) &= [Q_n^u(t) + a_n^u(t) - b_n^u(t)]^+, \\
Q_n^v(t+1) &= [Q_n^v(t) + a_n^v(t) - b_n^v(t)]^+, \\
Q_n(t+1) &= Q_n^v(t+1) + Q_n^u(t+1),
\end{aligned}$$

where $[a]^+ := \max\{a, 0\}$. We remind the reader that user traffic and adversary traffic are buffered in a single queue at each server and Q_n^u, Q_n^v's represent the amount of user or adversary packets in the queues. In this paper, we will impose the following assumption on the system and the space of the adversary injection policies.

Assumption 1. $\lim_{t\to\infty} \frac{Q(t)}{t}$, *and the time averages of arrivals and services exist*

[2]To avoid unnecessary complexity, we avoid formally defining the notion of adversary injection policy, but rather state it as the way that the adversarial dispatchers inject packets to the servers.

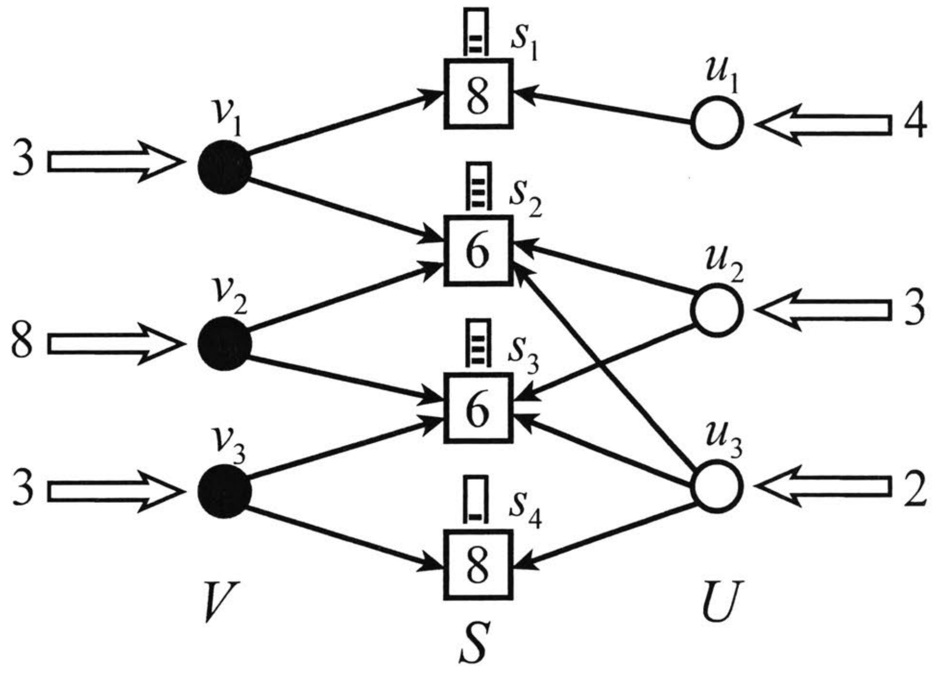


Figure 3-1: Illustration of our single-hop network model. User dispatchers are represented by hollow circles. Adversary dispatchers are represented by solid circles. Servers are represented by rectangles. The numerical values in the graph represent the traffic arrival rates to user/adversary dispatchers and the service rates of servers.

almost surely.

It can be shown that our model and the policies we will study satisfy the assumption provided that the random variables in the model satisfy some mild conditions [?]. We note that our results also hold under more general case, but require more careful and nuanced reasoning on convergence of random variables.

We give an illustration of our model in Figure 3-1. It will serve as the running example throughout our discussion of single-hop networks.

3.1.3 Problem Formulation

As we mentioned, the adversary conducts network DoS attack by controlling adversary dispatchers to send their available adversary packets to servers and seeking to prevent user packets from getting served. A network DoS attack is considered successful if the adversary manages to block certain portion of user packets from service. Formally, the *goal of the adversary* is that

$$\text{For some } n \in [N], \qquad \lim_{t\to\infty} \frac{\mathbb{E}[Q_n^u(t)]}{t} > 0. \tag{3.1}$$

By Little's law, (3.1) is equivalent to making the average delay experienced by user traffic grow unbounded (linearly) with time.

In an instance of network DoS attack, μ_n's and λ_l^u's can be seen as network while λ_m^v's can be viewed as the adversary's resource budget since it dictates how many packets the adversary dispatchers can inject to servers. We summarize the statistics and budget into vector forms as the service vector $\boldsymbol{\mu} = (\mu_1, \dots, \mu_N)$, user traffic arrival vector $\boldsymbol{\lambda}^u = (\lambda_1^u, \dots, \lambda_L^u)$ and adversary budget vector $\boldsymbol{\lambda}^v = (\lambda_1^v, \dots, \lambda_M^v)$.

Based on the above preliminaries, we formally define the network DoS attack problem.

Definition 4 (Network DoS Attack Problem). *Given a single-hop network with servers S, user dispatchers U, adversary dispatchers V. The network statistics and adversary budget vector are given by $\boldsymbol{\mu}, \boldsymbol{\lambda}^u$ and $\boldsymbol{\lambda}^v$, respectively. The Network DoS Attack Problem seeks an adversary injection policy under which there exists some server n with $\lim_{t\to\infty} \frac{\mathbb{E}[Q_n^u(t)]}{t} > 0$, i.e., the adversary achieves the goal with the policy. The problem is **feasible** if such an injection policy exists.*

Note that the definition of the Network DoS Attack problem only involves the means of the arrivals and services. We will demonstrate that the means are sufficient to characterize the problem.

Example: Consider the network in Figure 3-1, the network DoS attack problem is feasible. An injection policy that achieves the goal is that: v_1 injects all its traffic to s_2. v_2 injects half of its traffic to s_2 and the other half to s_3. v_3 injects all its traffic to s_3. Note that if v_1 and v_2 both injects all the traffic to s_2, this will cause the queue in s_2 to overflow, but will not achieve the goal. The (intuitive) explanation is that, as the user dispatchers are using JSQ, the user traffic from u_2 and u_3 will not be sent to s_2 (in equilibrium state). Since s_3 and s_4 have large enough capacities, there will not be any queue with $\lim_{t\to\infty} \frac{\mathbb{E}[Q_n^u(t)]}{t} > 0$.

3.2 Feasibility Region

In this section, we propose a necessary and sufficient condition on the network statistics and the adversary budget vector for the network DoS attack problem to be feasible. If we fix the network statistics, then the condition characterizes the feasibility region of the adversary. We begin by making some preliminary definitions.

For each subset of servers $S' \subseteq S$, we define $U_{S'}$ as the user dispatchers that only have connections to servers in S', i.e., $U_{S'} = \{u_l \mid S_{u_l} \subseteq S'\}$. We further define $\Delta(S')$ as

$$\Delta(S') = \sum_{s_n \in S'} \mu_n - \sum_{u_l \in U_{S'}} \lambda_l^u.$$

$\Delta(S')$ can be interpreted as the excess service rate of S' with respect to the user traffic generated by $U_{S'}$. Finally, for each S', we define the following linear program $LP(S')$ whose optimal value is denoted as $val(S')$.

$$\begin{aligned}
val(S') = \max & \sum_{m \in V} \sum_{n \in S'} f_{mn} && && (3.2)\\
\text{s.t.} & \sum_{n \in S'} f_{mn} \leq \lambda_m^v, && \forall m \in V && (3.3)\\
& \sum_{m \in V} f_{mn} \leq \mu_n, && \forall n \in S' && (3.4)\\
& f_{mn} = 0, && \text{if } n \notin S_{v_m} && \\
& f_{mn} \geq 0, && \forall m \in V. &&
\end{aligned}$$

$val(S')$ can be interpreted as the maximum amount of traffic that the adversary dispatchers can send to S' without exceeding the budget constraints of adversary dispatchers (Constraint (3.3)) or injecting to any server at a rate larger than its service (Constraint (3.4)). It will be clear soon that $val(S')$ represents the maximum meaningful capacity reduction that the adversary dispatchers can inflict on S', and it can be achieved by an stationary injection policy given by the solution to $LP(S')$. Relating $\Delta(S')$ and $val(S')$, we define the val-condition, which will be play a key role in the necessary and sufficient condition we propose.

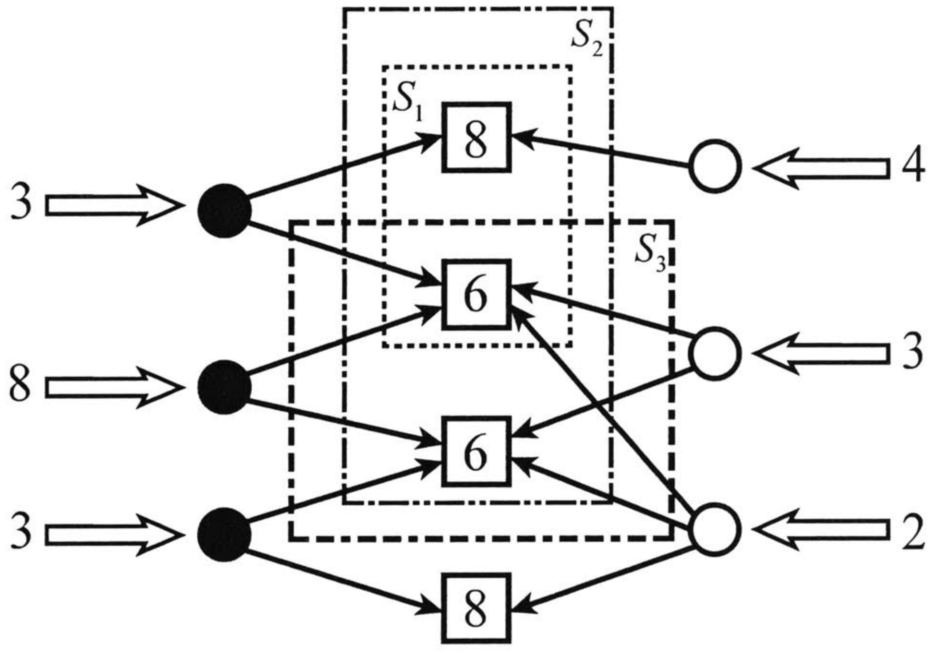


Figure 3-2: Illustration of the *val*-condition. Consider three subsets of servers S_1, S_2 and S_3 enclosed in dashed rectangles. $val(S_1) = 9, val(S_2) = 14, val(S_3) = 14$ and $\Delta(S_1) = 10, \Delta(S_2) = 13, \Delta(S_3) = 9$. Thus, S_2 and S_3 satisfies the *val*-condition while S_1 does not.

Definition 5 (The *val*-condition). *A subset of servers $S' \subseteq S$ satisfies the val-condition if $U_{S'}$ is non-empty and $val(S') > \Delta(S')$.*

We provide an illustration of the *val*-condition in Figure 3-2 (with the same network as in Figure 3-1).

Intuitively, if S' satisfies the *val*-condition, then it is possible for the adversary to make the residual capacity of S' be not enough for the incoming user traffic of $U_{S'}$, thus successfully blocking user traffic since the user traffic from $U_{S'}$ can only go to S'. It is then natural to consider that the network DoS attack problem is feasible if and only if there exists a subset of user dispatchers that satisfies the *val*-condition. We formalize the intuition in the following theorem.

Theorem 5. *The network DoS problem is feasible if and only if there exists a subset of servers $S' \subseteq S$ that satisfies the val-condition.*

Proof. The proof is divided into two parts. In the first part, we prove the sufficiency of the *val*-condition by showing that, if there exists $S' \subseteq S$ that satisfies the condition, then the stationary injection policy induced by $LP(S')$ achieves the goal of the adversary. In the second part, we prove the necessity of the *val*-condition by starting from any given adversary injection policy that achieves the goal and taking time averages, which will lead to establishing that some S' satisfies the *val*-condition.

Sufficiency: If there exists a $S' \subseteq S$ such that $U_{S'}$ is non-empty and $val(S') > \Delta(S')$, we denote the solution to $LP(S')$ as $\{f^*\}_{mn}$. Consider the following randomized injection policy for adversary: at every time slot, each adversary traffic dispatcher m that has connection to S' injects its traffic to server $n \in S'$ with probability $f^*_{mn}/\sum_{n'} f^*_{mn'}$; other adversary dispatchers inject the traffic arbitrarily. We proceed to show that such policy achieves the goal, i.e., under the injection policy, there exists a server n such that $\lim_{t\to\infty} \frac{\mathbb{E}[Q^u_n(t)]}{t} > 0$.

First, examining the servers in S', we have

$$\frac{\sum_{n\in S'} Q_n(t)}{t} \geq \frac{\sum_{n\in S'}\sum_{i=1}^{t-1} a_n(i) - \sum_{n\in S'}\sum_{i=1}^{t-1} b_n(i)}{t}$$
$$= \sum_{n\in S'} \left(\frac{\sum_{i=1}^{t-1} a^u_n(i)}{t} + \frac{\sum_{i=1}^{t-1} a^v_n(i)}{t} - \frac{\sum_{i=1}^{t-1} b_n(i)}{t} \right)$$
$$= \sum_{i=1}^{t-1} \left(\frac{\sum_{n\in S'} a^u_n(i)}{t} + \frac{\sum_{n\in S'} a^v_n(i)}{t} - \frac{\sum_{n\in S'} b_n(i)}{t} \right)$$

Let t goes to infinity in the above inequalities. By law of large numbers [?], the limits exist with probability one.[3] Therefore, on each sample path (except a set of measure

[3]For notational brevity, we will often use the same symbol for both random variables and their realizations on sample paths, e.g. $Q_n(t)$ as $Q_n(t, \omega)$

zero), we have

$$\begin{aligned}
&\lim_{t\to\infty} \frac{\sum_{n\in S'} Q_n(t)}{t} \\
\geq &\lim_{t\to\infty} \sum_{i=1}^{t-1} \left(\frac{\sum_{n\in S'} a_n^u(i)}{t} + \frac{\sum_{n\in S'} a_n^v(i)}{t} - \frac{\sum_{n\in S'} b_n(i)}{t} \right) \\
\geq &\sum_{l\in U_{S'}} \lambda_l^u + \sum_{m\in V} \sum_{n\in S'} f_{mn}^* - \sum_{n\in S'} \mu_n \\
= &val(S') - \Delta(S') > 0. \qquad (3.5)
\end{aligned}$$

From Inequality (3.5), we claim the following, which will lead to the first part of the proof.

Claim 1. *With probability one, there exists $n \in S'$ such that $\lim_{t\to\infty} \frac{Q_n^u(t)}{t} > 0$.*

Proof of Claim 1: We prove the claim by showing that we can find such n on each sample path (except a set of measure zero). Fixing an arbitrary sample path, by (3.5), there exists an $n_1 \in S'$ such that $\lim_{t\to\infty} \frac{Q_{n_1}(t)}{t} > 0$. If $\lim_{t\to\infty} \frac{Q_{n_1}^u(t)}{t} > 0$, then we finish the proof. Otherwise, we have that $\lim_{t\to\infty} \frac{Q_{n_1}^v(t)}{t} > 0$ and $\lim_{t\to\infty} \frac{Q_{n_1}^u(t)}{t} = 0$. Since server n_1 is overloaded and its service discipline is FCFS, $\lim_{t\to\infty} \frac{Q_{n_1}^u(t)}{t} = 0$ implies that $\lim_{t\to\infty} \frac{\sum_{i=1}^{t-1} a_{n_1}^u(i)}{t} = 0$. This means that the time-average traffic rate from $U_{S'}$ to n_1 is zero. Now, we consider the subset of servers $S_1' = S' \backslash \{n_1\}$. Note that since $\sum_{m\in V} f_{mn_1}^* \leq \mu_{n_1}$, we have that $\sum_{m\in V} \sum_{n\in S_1'} f_{mn}^* \geq val(S') - \mu_{n_1}$. It follows that

$$\begin{aligned}
&\lim_{t\to\infty} \frac{\sum_{n\in S_1'} Q_n(t)}{t} \\
\geq &\lim_{t\to\infty} \sum_{n\in S'} \left(\frac{\sum_{i=1}^{t-1} a_n^u(t)}{t} + \frac{\sum_{i=1}^{t-1} a_n^v(t)}{t} - \frac{\sum_{i=1}^{t-1} b_n(t)}{t} \right) \\
\geq &\sum_{u_l\in U_{S'}} \lambda_l^u + \sum_{m\in V} \sum_{n\in S_1'} f_{mn}^* - \sum_{n\in S_1'} \mu_n \\
\geq &[val(S') - \mu_n] - [\Delta(S') - \mu_n] \\
> &0,
\end{aligned}$$

where the second inequality follows from that $\lim_{t\to\infty} \frac{\sum_{i=1}^{t-1} a^u_{n_1}(i)}{t} = 0$. Therefore, we can repeat the argument above, that there must exist another $n_2 \in S'_1$ such that $\lim_{t\to\infty} \frac{Q_{n_2}(t)}{t} > 0$. And if $\lim_{t\infty} \frac{Q^u_{n_2}(t)}{t} = 0$, then again, it implies that the time-average traffic rate from $U_{S'}$ to n_2 is also zero. By repeating such argument, we will arrive at one of the two following situations: (i). we find a queue find a queue n_i such that $\lim_{t\to\infty} \frac{Q^u_{n_i}(t)}{t} > 0$; (ii). we arrive at S'_k that is a singleton set $\{n_k\}$ with $\lim_{t\to\infty} \frac{Q_{n_k}(t)}{t} > 0$. Furthermore, $\lim_{t\to\infty} \frac{\sum_{i=1}^{t-1} a^u_{n_j}(i)}{t} = 0$ for $j = 1, \ldots, k-1$. This implies that $\lim_{t\to\infty} \frac{Q^u_{n_k}(t)}{t} > 0$ as $U_{S'}$ is non-empty and the user traffic in $U_{S'}$ must go to somewhere in S'. Therefore, there exists some n such that $\lim_{t\to\infty} Q^u_n(t)/t > 0$ with probability 1.

From the proof of Claim 1, it is easy to see that the claim can be strengthened as, with probability 1, for some $\epsilon > 0$, there exists $n \in S'$ such that $\lim_{t\to\infty} Q^u_n(t)/t > \epsilon$. Since S' is finite, we have there exists a $n \in S'$ such that $\lim_{t\to\infty} Q^u_n(t)/t > \epsilon$ with probability at least $1/|S'|$ where $|S'|$ denotes the cardinality of S'. Hence, there must exist an n with $\lim_{t\to\infty} \mathbb{E}[Q^u_n(t)]/t > 0$. Thus, we have shown that the randomized policy achieves the goal if there exists an U' that satisfies the *val*-condition.

Necessity: Suppose that there exists an injection policy that achieves the goal, then there must exists a sample path on which $\lim_{t\to\infty} \frac{Q^u_n(t)}{t} > 0$ for some n. Under one such sample path, let $S' \in S$ be the subset of servers that ended up getting overflowed under such policy, i.e., $\forall n \in S'$, $\lim_{t\to\infty} \frac{Q_n(t)}{t} > 0$ and $\forall n \notin S'$, $\lim_{t\to\infty} \frac{Q_n(t)}{t} = 0$. Since the queues in S' grow linearly with time while the queues in $S\backslash S'$ do not, eventually queues in S' will be longer than those in $S\backslash S'$. Therefore, for every user dispatcher u_l that has connection to servers that are not in S' the time average arrival rates from u_l to servers in S' are all zero, by the property of JSQ. Formally, we have

$$\lim_{t\to\infty} \sum_{n\in S'} \sum_{i=1}^{t-1} \frac{a^u_{ln}(i)}{t} = 0, \quad \forall u_l \ s.t.\ S_{u_l} \cap (S\backslash S') \neq \emptyset. \tag{3.6}$$

We now claim in the following that S' satisfies the *val*-condition. By establishing the claim, we prove the necessity part of the theorem.

Claim 2. *S' satisfies the val-condition.*

Proof of Claim 2: Recall that the set of user dispatchers that only have connections to servers in S' is denoted as $U_{S'}$. We first prove that $U_{S'}$ is not empty. For if not, suppose $U_{S'}$ is empty, then by Equation (3.6), the time average rates from all user dispatchers to S' are zero, i.e., the user dispatchers inject all their traffic to the non-overloaded servers $S \backslash S'$, and thus the goal would not be achieved.

We now proceed to demonstrate that $val(S') > \Delta(S')$. Note that for every $n \in S'$, $\lim_{t\to\infty} Q_n(t)/t > 0$. It follows that the expected sum arrival rate at server n must be greater than its service rate. Therefore, for each $n \in S'$,

$$\lim_{t\to\infty}\left(\frac{\sum_{i=1}^{t-1} a_n^u(i)}{t} + \frac{\sum_{i=1}^{t-1} a_n^v(i)}{t} - \frac{\sum_{i=1}^{t-1} b_n(i)}{t}\right)$$
$$= \lim_{t\to\infty}\left(\frac{\sum_{i=1}^{t-1} a_n^u(i)}{t} + \frac{\sum_{i=1}^{t-1} a_n^v(i)}{t}\right) - \mu_n > 0.$$

It follows that for each $n \in S'$

$$\lim_{t\to\infty}\frac{\sum_{i=1}^{t-1} a_n^u(i)}{t} + \min\left(\lim_{t\to\infty}\frac{\sum_{i=1}^{t-1} a_n^v(i)}{t} - \mu_n, 0\right) \geq 0.$$

Based on this, we define a set of $\{f\}_{mn}$ as:

$$f_{mn} = \lim_{t\to\infty}\frac{\sum_{i=1}^{t-1} a_{mn}^v(i)}{t}, \quad \text{if } \lim_{t\to\infty}\frac{\sum_{i=1}^{t-1} a_n^v(i)}{t} \leq \mu_n,$$

and otherwise,

$$f_{mn} = \left(\lim_{t\to\infty}\frac{\sum_{i=1}^{t-1} a_{mn}^v(i)}{t} / \lim_{t\to\infty}\frac{\sum_{i=1}^{t-1} a_n^v(i)}{t}\right)\cdot \mu_n.$$

It is easy to verify that $\{f\}_{mn}$ satisfies the constraints of $LP(S')$. Furthermore,

$$\sum_{n\in S'}\lim_{t\to\infty}\frac{\sum_{i=1}^{t-1} a_n^u(i)}{t} + \sum_{n\in S'}\sum_{m\in V} f_{mn} - \sum_{n\in S'}\mu_n > 0.$$

Here, the strict inequality holds since $U_{S'}$ is non-empty. It follows that

$$0 < \sum_{u_l \in U_{S'}} \lambda_l^u - \sum_{n \in S'} \mu_n + \sum_{n \in S'} \sum_{m \in V} f_{mn} \qquad \leq val(S') - \Delta(S'),$$

which implies that S' follows the *val*-condition.

The necessity of *val*-condition follows directly from Claim 2. $\square$

Based on Theorem 5, we have the following corollary. It states that to check the feasibility of the network DoS problem, we only need to check the subsets of servers induced by subsets of user dispatchers.

Corollary 2. *The network DoS problem is feasible if and only if there exists a non-empty subset of user dispatchers $U' \subseteq U$, such that $S_{U'} = \bigcup_{u_l \in U'} S_{u_l}$ satisfies the val-condition.*

Proof. From the definition of *val* function, we have that if there exists a subset of servers S' that satisfies the *val*-condition, then $U_{S'}$ satisfies the condition in Corollary 2. Conversely, if $U' \subseteq U$ satisfies the condition in the corollary, then $S_{U'} \subseteq S$ satisfies the *val*-condition by design. $\square$

Remark: For an adversary injection policy, it is *optimal* if the adversary achieves the goal under such policy whenever the network DoS attack problem is feasible (i.e., it achieves the feasibility region.); it is *oblivious* if the policy does not rely on knowledge of the network statistics $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}^u$ or budget vector $\boldsymbol{\lambda}^v$. The proof of Theorem 5 already gives us an stationary-randomized adversary injection policy that is optimal. However, the policy is not oblivious since it involves solving LP's which depend heavily on network statistics. Since in practice the adversary often does not have such knowledge, if the feasibility region could not be achieved by any oblivious policy, then it would not serve as a tight characterization of the fundamental limit of network DoS attack. We bridge this gap by showing in the sequel is that, an optimal and oblivious policy does exist, which justifies the validity of the feasibility condition in practice.

3.3 Preliminary Results

In this section, we introduce the key preliminary results that lay the foundation of the adversary injection policy that we propose.

3.3.1 Queue Length Behavior Under JSQ

Consider a server farm with set of servers $S = \{s_1, \ldots, s_N\}$ and set of traffic dispatchers $U = \{u_1, \ldots, u_L\}$ (There is no adversary dispatcher.). The traffic dispatchers route incoming traffic to servers following the JSQ rule, with ties broken arbitrarily. Similar as in our model, the service rate of each server s_n at every time slot is a random variable (i.i.d across time) with mean μ_n, and the arrival at each dispatcher u_l is a random variable (i.i.d across time) with mean λ_l. The μ_n's determine the throughput region of the network, i.e., the set of λ_l's that are supportable under some routing policy. It is well known that JSQ is throughput-optimal in the sense that using JSQ, the server farm can support the incoming traffic as long as λ_l's lie in the throughput region [?]. It is also intuitively understood that when the arrival rates are outside the throughput region, JSQ achieves graceful failure such that the queues grow with time in a balanced manner. The following result makes this precise.

Consider the following optimization problem $\mathcal{P}$

$$\min \sum_n r_n^2 \tag{3.7}$$

$$\text{s.t.} \sum_{n \in S_{u_l}} \lambda_{ln} = \lambda_l, \qquad \forall l \in U \tag{3.8}$$

$$r_n \geq \sum_{l: n \in S_{u_l}} \lambda_{ln} - \mu_n, \qquad \forall n \in S \tag{3.9}$$

$$\lambda_{ln} = 0, \qquad \forall n \notin S_{u_l} \tag{3.10}$$

$$r_n, \lambda_{ln} \geq 0, \qquad \forall l, n. \tag{3.11}$$

Note that $\mathcal{P}$ is a convex optimization problem, and hence has a unique optimal solution. We have the following proposition.

Proposition 6. *Let $\mathbf{r}^* = (r_1^*, \dots, r_n^*)$ be the optimal solution to the optimization problem $\mathcal{P}$ associated with the server farm. The queue lengths at the servers satisfy that for any $\delta > 0$, for each n,*

$$\lim_{t\to\infty} \mathbb{P}\left\{ \left| \frac{Q_n(t)}{t} - r_n^* \right| < \delta \right\} = 1.$$

Proof. The proposition follows by applying the results from two papers by Shah and Wischik [47,48]. We defer the details to the appendix. □

Proposition 6 establishes that under JSQ, the growth rate of queue lengths (over time) converges to the optimal solution to $\mathcal{P}$ in probability. It characterizes the queue length behavior under JSQ in both under-load and over-load regimes. In the former case, the solution is the zero vector, which implies that queues do not grow with time; in the latter case, the optimal solution is the "most balanced" overflow rate that is achievable, and the overflow rate under JSQ converges to that.

Further, we consider the optimization problem $\mathcal{P}'$, which is a modified version of $\mathcal{P}$ that removes the non-negativity constraints on r_n and replace constraint (3.9) with equality:

$$\begin{aligned} &\min \sum_n r_n^2 \\ \textbf{s.t.} &\sum_{n\in S_{u_l}} \lambda_{ln} = \lambda_l, \qquad \forall l \in U \\ &r_n = \sum_{l:n\in S_{u_l}} \lambda_{ln} - \mu_n, \qquad \forall n \in S \\ &\lambda_{ln} = 0, \qquad \forall n \notin S_{u_l} \\ &\lambda_{ln} \geq 0, \qquad \forall l, n. \end{aligned}$$

$\mathcal{P}'$ is also convex and has unique optimal solution. If we consider an (virtual) alternate network dynamics that allow the queues to go negative, i.e., $Q_n(t+1) = Q_n(t) - b_n(t) + a_n(t)$, then we can link the queue length behavior under JSQ in such alternate dynamics with the solution to $\mathcal{P}'$.[4]

Proposition 7. *Let $\tilde{r}^* = (\tilde{r}_1^*, \ldots, \tilde{r}_n^*)$ be the optimal solution to the optimization problem $\mathcal{P}'$ associated with the server farm having the alternate dynamics . The queue lengths at the servers satisfy that for any $\delta > 0$, for each n,*

$$\lim_{t\to\infty} \mathbb{P}\left\{ \left| \frac{Q_n(t)}{t} - \tilde{r}_n^* \right| < \delta \right\} = 1. \tag{3.12}$$

Proof. The proof is the same as Proposition 6. □

By expanding the limit expression (3.12), we obtain the following corollary.

Corollary 3. *Under the alternate dynamics, if $\min_n \tilde{r}_n^* > 0$, then $\forall \epsilon > 0$, there exists a $T_\epsilon > 0$ such that for all $t \geq T_\epsilon$,*

$$\mathbb{P}\left\{ \forall n,\ Q_n(t) \geq \frac{\tilde{r}_n^* t}{2} \right\} \geq 1 - \epsilon.$$

Proof. Take $\delta = \frac{1}{2}\min_n \tilde{r}_n^* > 0$, by Proposition 7, for each n,

$$\lim_{t\to\infty} \mathbb{P}\left\{ \frac{Q_n(t)}{t} \geq \frac{\tilde{r}_n^*}{2} \right\} = 1$$

Hence, fix a $\epsilon > 0$, for each n, there exists a $T_{n,\epsilon}$ such that for all $t \geq T_{n,\epsilon}$,

$$\mathbb{P}\left\{ \frac{Q_n(t)}{t} \geq \frac{\tilde{r}_n^*}{2} \right\} \geq 1 - \frac{\epsilon}{N} \implies \mathbb{P}\left\{ \frac{Q_n(t)}{t} < \frac{\tilde{r}_n^*}{2} \right\} \leq \frac{\epsilon}{N}.$$

Let $T_\epsilon = \max_n T_{n,\epsilon}$. By union bound, we have for all $t \geq T_\epsilon$,

$$\mathbb{P}\left\{ \exists n, \frac{Q_n(t)}{t} < \frac{\tilde{r}_n^*}{2} \right\} \leq \epsilon.$$

Taking the complement, it follows that

$$\mathbb{P}\left\{ \forall n,\ \frac{Q_n(t)}{t} \geq \frac{\tilde{r}_n^*}{2} \right\} \geq 1 - \epsilon.$$

[4]Under the alternate dynamics, JSQ specifies that the dispatchers route traffic to servers with the minimum queue length value.

□

The two optimization problem $\mathcal{P}$ and $\mathcal{P}'$ differ only in the non-negativity constraints of r_n's. Our next result shows that their optimal solutions are identical under certain condition. For two vectors $\boldsymbol{r}$ and $\tilde{\boldsymbol{r}}$, we write $\boldsymbol{r} > \tilde{\boldsymbol{r}}$ if $\forall n,\ r_n > \tilde{r}_n$; $\boldsymbol{r} \geq \tilde{\boldsymbol{r}}$ if $\forall n,\ r_n \geq \tilde{r}_n$; $\boldsymbol{r} = \tilde{\boldsymbol{r}}$ if $\forall n,\ r_n = \tilde{r}_n$.

Proposition 8. *Let $\boldsymbol{r}^*$ be the optimal solution to $\mathcal{P}$ and $\tilde{\boldsymbol{r}}^*$ be the optimal solution to $\mathcal{P}'$. For any n, $r_n^* > \mathbf{0}$, then $r_n^* = \tilde{r}_n^*$.*

Proof. We first show a simplified version of Proposition 8: if $\boldsymbol{r}^* > \mathbf{0}$, then $\boldsymbol{r}^* = \tilde{\boldsymbol{r}}^*$, and then extend the proof to the original version. We prove the simplified version by establishing two claims: (i). Under the condition that $\boldsymbol{r}^* > \mathbf{0}$, if $\tilde{\boldsymbol{r}}^* \geq \mathbf{0}$, then $\boldsymbol{r}^* = \tilde{\boldsymbol{r}}^*$ and (ii). Under the condition that $\boldsymbol{r}^* > \mathbf{0}$, $\tilde{\boldsymbol{r}}^* \geq \mathbf{0}$. These two claims combined yield the proposition.

We start with the first claim. Observe that if $\boldsymbol{r}_n^* > 0$ in the optimal solution to $\mathcal{P}$, then the corresponding constraint (3.9) must be satisfied with equality, since otherwise we can decrease $\boldsymbol{r}_n^*$ and obtain a better solution. Therefore, $\boldsymbol{r}^*$ must be feasible to $\mathcal{P}'$. Conversely, if some $\tilde{\boldsymbol{r}} \geq \mathbf{0}$ is feasible to $\mathcal{P}'$, then it must also be feasible to $\mathcal{P}$. Combining these, we have under the condition that $\boldsymbol{r}^* > \mathbf{0}$, if $\tilde{\boldsymbol{r}}^* \geq \mathbf{0}$, then $\boldsymbol{r}^*$ is feasible to $\mathcal{P}'$ and $\tilde{\boldsymbol{r}}^*$ is feasible to $\mathcal{P}$. Since both $\mathcal{P}$ and $\mathcal{P}'$ have unique optimal solution and they have the same objective function, we have $\boldsymbol{r}^* = \tilde{\boldsymbol{r}}^*$. We now proceed to establish the second claim. If there is an n such that $\tilde{r}_n^* < 0$, we consider a dispatcher l such that $n \in S_{u_l}$. We claim that for all $n' \in S_{u_l}$, $\tilde{r}_{n'}^* < 0$. Since otherwise if there is a $n' \in S_{u_l}$ with $\tilde{r}_{n'}^* \geq 0$, we can increase λ_{ln} and decrease $\lambda_{ln'}$ and form a vector $\tilde{\boldsymbol{r}}'$ with $||\tilde{\boldsymbol{r}}'||^2 < ||\tilde{\boldsymbol{r}}^*||^2$, which contradicts the optimality of $\tilde{\boldsymbol{r}}^*$. We repeat this argument for other user dispatchers that have connections to S_{u_l}, until we arrive at a subset of servers S' that satisfies: (i). $\tilde{r}_n^* < 0$ for all $n \in S'$ and (ii). there does not exist a user dispatcher that has connection to both servers in S' and servers in $S \backslash S'$. This implies that in the optimization problem $\mathcal{P}$, it is feasible to have $r_n = 0$ for all $n \in S'$ since the constraints for $n \in S'$ and $n \in S \backslash S'$ do not interfere with one another. Hence, by setting the entries in $\boldsymbol{r}^*$ that correspond to all

$n \in S'$ to zero, we obtain a better solution to $\mathcal{P}$, which contradicts the optimality of $\boldsymbol{r}^*$. Hence, we prove the second claim, and conclude the proof of: if $\boldsymbol{r}^* > \boldsymbol{0}$, then $\boldsymbol{r}^* = \tilde{\boldsymbol{r}}^*$.

Now, for an arbitrary n such that $r_n^* > 0$, again, consider some dispatcher l such that $n \in S_{u_l}$. We have $r_{n'}^* > 0$ for all $n' \in S_{u_l}$, since otherwise one can decrease λ_{ln} and increase $\lambda_{ln'}$ for some n' with $r_{n'}^* \leq 0$ and obtain a better $\boldsymbol{r}$ than $\boldsymbol{r}^*$. Repeating this argument, by a similar reasoning as above, we will arrive at a subset of servers S' that satisfies $r_n^* > 0$ for all $n \in S'$ and there is no user dispatcher that has connection to both S' and $S \backslash S'$. Therefore, we can decompose $\mathcal{P}$ into two parts that correspond to S' and $S \backslash S'$ respectively. Applying the previously proved simplified version to the part of S', we have $r_{n'}^* = \tilde{r}_{n'}^*$ for all $n' \in S'$, which include the n we start with. Hence, we conclude the proof of the proposition. □

3.3.2 Transience of Markov Chain

In this section, we introduce a sufficient condition for transience of countable-state Markov Chain. It will play a key role in establishing that the policy we propose achieves the goal of the network DoS attack problem. The condition, stated in Lemma 7, is adapted from Theorem 2.2.7 in [49].

Lemma 7. *Let $\mathcal{L}$ be an irreducible countable-state discrete-time Markov Chain with state space $\mathcal{A}$. Let $X(t)$ denote the state of the chain at time t. The chain $\mathcal{L}$ is transient, if there exist a non-negative function (Lyapunov Function) $f(\alpha), \alpha \in \mathcal{A}$, a positive integer k, and $\epsilon > 0, \gamma > 0$, such that, setting $A_\gamma = \{\alpha : f(\alpha) > \gamma\} \neq \emptyset$, the following conditions hold:*

1. *$\mathbb{E}[f(X(t+k)) - f(X(t)) \mid X(t) = \alpha_i] \geq \epsilon$, for all t, and $\alpha_i \in A_\gamma$.*
2. *for some $d > 0$, the inequality $f(\alpha_i) - f(\alpha_j) < -d$ implies that the one-step transition probability from α_i to α_j is zero.*[5]

[5]The original version of Theorem 2.2.7 in [49] states this condition as "the inequality $|f(\alpha_i) - f(\alpha_j)| < d$ implies that the one-step transition probability from α_i to α_j is zero". One can easily show from the proof of Theorem 2.2.7 in [49] that the original condition can be relaxed to the one we state in this paper.

Furthermore, let $X(0)$ be a state such that $f(X(0)) \in A_\gamma$. Define the stopping time $\tau = \inf\{t \geq 1 : f(X(t)) \leq \gamma \mid X(0)\}$. If the above two conditions hold, then $\mathbb{P}\{\tau = \infty\} > 0$.

The lemma can be interpreted as a converse of the Foster-Lyapunov theorem. It states that, if we can find a Lyapunov function on the state space such that in conditioning on a subset of states, the Lyapunov function has positive k-slot drift, then the Markov Chain is transient.

3.3.3 Monotonicity Property of JSQ

We present a sample path-wise bound regarding the queue length vector of JSQ server farm with the alternate dynamics. Consider a server farm with servers $\{s_1, \ldots, s_N\}$ and dispatchers $\{u_1, \ldots, u_L\}$, where all dispatchers use JSQ routing. The arrivals and services at each time slot are upper bounded by C. We assume that the queues evolve under the alternate dynamics (as in Proposition 7), which means that all the realized services equal the offered services and the queue lengths can become negative. Consider a sample path of such system. Observe that if we are given an initial queue length vector $\boldsymbol{Q}(0) = \{Q_1(0), \ldots, Q_n(0)\}$, sequence of arrivals at user dispatchers $(\boldsymbol{\lambda}^u(0), \boldsymbol{\lambda}^u(1), \ldots)$ where each $\boldsymbol{\lambda}^u(t) = (\lambda_1^u(t), \ldots, \lambda_L^u(t))$ specifies the arrivals at time t, sequence of offered services at servers $(\boldsymbol{b}(0), \boldsymbol{b}(1), \ldots)$ where each $\boldsymbol{b}(t) = (b_1(t), \ldots, b_N(t))$ specifies the offered services at time t, and certain tie-breaking rule, then the queue length vector at each future time slot can be fully determined. Therefore, consider two JSQ server farms whose traffic arrivals and services are identical random variables, one has initial queue length vector $\boldsymbol{Q}(0)$ and the other has $\tilde{\boldsymbol{Q}}(0)$. We can couple the (random) queue length vectors at time slot t of the two system in a sample path-wise manner. At each sample path with some common sequences of arrivals $(\boldsymbol{\lambda}^u(0), \boldsymbol{\lambda}^u(1), \ldots)$ and services $(\boldsymbol{b}(0), \boldsymbol{b}(1), \ldots)$, let $\boldsymbol{Q}(t)$ and $\tilde{\boldsymbol{Q}}(t)$ be the (deterministic) queue length vectors $\boldsymbol{Q}(0)$ and $\tilde{\boldsymbol{Q}}(\mathbf{0})$ evolve into at t under the sequence, respectively. Since the arrivals and services of the two systems are identical random variables, the above procedure forms a coupling. If we can prove some relation between deterministic queue length vectors $\boldsymbol{Q}(t)$ and $\tilde{\boldsymbol{Q}}(t)$, we will obtain that the two random queue length vectors satisfy such relation in a stochastic sense. The following proposition gives one such relation.

Proposition 9. *If $\boldsymbol{Q}(0) \geq \tilde{\boldsymbol{Q}}(0)$, then at each time t, for all n,*

$$Q_n(t) \geq \tilde{Q}_n(t) - N_1 LC,$$

where $N_1 = N! + 1$. The result holds for arbitrary tie-breaking rules that the two systems use.

Proof. We defer rigorous proof to Appendix 3.8.2 and gives some intuition here. Suppose that the system is performing JSQ with the finest granularity, i.e., for each packet that arrives at some dispatcher, the dispatcher sends the packet to the server with the shortest queue (among the ones that it is connected to) and the queue lengths are updated immediately afterwards. Then, one can actually show that under certain tie-breaking, $Q_n(t) \geq \tilde{Q}_n(t)$ for all n, t. The argument is that such relation is invariant through any packet transmission, provided that the ties are breaking appropriately. The additional MLC factor in Proposition 9 accounts for different tie-breaking rules, and that the JSQ in our model does not need to be at such fine granularity. □

3.4 Optimal Oblivious Adversary Injection Policy

In this section, we design an optimal oblivious adversary injection policy, that achieves the goal whenever it is feasible and does not require knowledge of network statistics. From the proof of Theorem 5, we observe that intuitively an optimal policy should be able to identify a subset of servers that satisfies the *val*-condition and appropriately allocate adversary dispatchers' traffic to the subset of servers without over-expending budget on any single server (c.f. constraint (3.9)). For an optimal policy to be oblivious, it needs to achieve the two aforementioned tasks based solely on queue-length information rather than network statistics. Before introducing such policy, we first present an intermediate policy that is optimal but semi-oblivious, in the sense that it achieves the second task without relying on network statistics, i.e., given a subset of servers that satisfies the *val*-condition, the policy achieves the goal decides the adversary injection based on queue length information only. The policy, called "Target-JSQ policy", brings out a key idea and paves the way to the optimal oblivious policy. Therefore, we first present and analyze the Target-JSQ policy.

3.4.1 Target-JSQ Policy

As its name suggests, the Target-JSQ policy works by identifying a subset of servers that satisfies the *val*-condition, and then making all the adversary dispatchers that have connection to that subset send packets to the servers following JSQ rule. Formally, let S' be a subset that satisfies the *val*-condition. For all the adversary dispatchers v_m such that $S_{v_m} \cap S' \neq \emptyset$, at each time slot, v_m send its packets to the servers in $S_{v_m} \cap S'$ with the shortest queue. Other adversary dispatchers send packets arbitrarily (or do not send packets at all). Theorem 6 establishes the optimality of the Target-JSQ policy.

Theorem 6. *Suppose $S' \subseteq S$ satisfies the val-condition and all the adversary dispatchers that have connections to S' inject traffic (only) to S' according to the JSQ rule, then we have*

$$\exists n \in S', \lim_{t \to \infty} \frac{\mathbb{E}[Q_n^u(t)]}{t} > 0.$$

Proof. Define $V_{S'}$ as the subset of adversary dispatchers that have connections to S', i.e., $V_{S'} = \{v_m \mid S_{v_m} \cap S' \neq \emptyset\}$, and recall that $U_{S'}$ is defined to be the subset of user dispatchers that only have connection to S', i.e., $U_{S'} = \{u_l \mid S_{u_l} \subseteq S'\}$. Since S' satisfies the *val*-condition, $U_{S'}$ is non-empty. If $V_{S'}$ is empty, then $\Delta(U') < 0$, which means that the total rate of incoming user traffic to S' is greater than the total service rate of S'. Hence, the theorem vacuously holds. Therefore, we can assume that $V_{S'}$ is not empty. For simplicity, we consider the case where the adversary dispatchers in $V_{S'}$ inject packets to S' according to the JSQ rule, and other adversary dispatchers send nothing to the servers. The argument applies to the case where other adversary dispatchers inject arbitrarily as well. Now, we study the system formed by user dispatchers U, adversary dispatchers $V_{S'}$ and servers S, with $V_{S'}$ only have connections to S'. Note that the system is a server farm where all the dispatchers $(U \cup V_{S'})$ employ JSQ routing. Therefore, the queue length growth rates observe Proposition 6. Let $\mathcal{P}$ be the optimization problem in Proposition 6 that associates with the system and $\boldsymbol{r}^*, \boldsymbol{\lambda}^*$ be the optimal solution to $\mathcal{P}$. We will use λ^*_{ln} and λ^*_{mn} to denote the component in $\boldsymbol{\lambda}^*$ that correspond to user dispatcher u_l and adversary dispatcher v_m, respectively.

For the subset of servers S', the total incoming rate of adversary traffic equals $\sum_{m \in V_{S'}} \lambda^v_m$, which by definition, is greater than or equal to $val(S')$. The total incoming rate of user traffic is at least $\sum_{u_l \in U_{S'}} \lambda^u_l$. Since S' satisfies the val-condition, we have $\sum_{m \in V_{S'}} \lambda^v_m + \sum_{u_l \in U_{S'}} \lambda^u_l > \sum_{s_n \in S'} \mu_n$. It follows by summing up constraints (3.9) of $\mathcal{P}$ over all $n \in S'$ that $\boldsymbol{r}^*$ must have positive entry. Let $\tilde{S}'$ be the set of servers in S' whose corresponding entry is positive in $\boldsymbol{r}^*$, i.e., $\tilde{S}' = \{s_n \in S' \mid \boldsymbol{r}^*_n > 0\}$. By the above reasoning, $\tilde{S}' \neq \emptyset$. We will show in the following lemma that there must exist user dispatcher u_l such that $S_{u_l} \subseteq \tilde{S}'$, which means that the user traffic from u_l can only go to servers in $\tilde{S}'$.

Lemma 8. *There exists user dispatcher u_l such that $S_{u_l} \subseteq \tilde{S}'$.*

Proof. Assume for the sake of contradiction that there is no such user dispatcher u_l, that is, every user dispatcher has connection to $S \backslash \tilde{S}'$. Under this condition, we first establish the following claim.

Claim 3. *The optimal solution λ^* must satisfies that $\lambda^*_{ln} = 0, \lambda^*_{mn} = 0$ for all $n \in \tilde{S}'$, user dispatcher u_l and adversary dispatcher v_m that have connection to server in $S \backslash \tilde{S}'$.*

Proof of Claim 3: If the claim does not hold, then we can decrease some positive λ^*_{ln} (or λ^*_{mn}) by a small amount $\delta > 0$ and add to $\lambda^*_{ln'}$ (or $\lambda^*_{mn'}$) with $n' \in S \backslash \tilde{S}'$. This will result in that $\boldsymbol{r}^*_n > 0$ decreases by δ and $\boldsymbol{r}^*_{n'} = 0$ increases by δ, thereby obtaining a $\boldsymbol{r}$ with a smaller value of $\sum_n \boldsymbol{r}^2_n$, which contradicts the optimality of $\boldsymbol{r}^*$.

Based on Claim 3, we proceed to show the following.

Claim 4. *If there is no user dispatcher u_l with $S_{u_l} \subseteq \tilde{S}'$, then*

$$\sum_{m \in V_{S'}} \sum_{n \in S' \setminus \tilde{S}'} \lambda^*_{mn} \geq val(S') - \sum_{n \in \tilde{S}'} \mu_n, \tag{3.13}$$

Proof of Claim 4: Let $\tilde{V}_{S'}$ be the subset of adversary dispatchers that only have connections in $\tilde{S}'$. Note that $\tilde{V}_{S'} \subseteq V_{S'}$. By Claim 3, we have $\lambda^*_{mn} = 0$ for all $m \notin \tilde{V}_{S'}, n \in \tilde{S}'$. Therefore,

$$\begin{aligned}\sum_{m \in V_{S'}} \sum_{n \in S' \setminus \tilde{S}'} \lambda^*_{mn} &\geq \sum_{m \notin \tilde{V}_{S'}} \sum_{n \in S' \setminus \tilde{S}'} \lambda^*_{mn} \\ &= \sum_{m \notin \tilde{V}_{S'}} \sum_{n \in S'} \lambda^*_{mn} = \sum_{m \notin \tilde{V}_{S'}} \lambda^v_m. \end{aligned} \tag{3.14}$$

Now, recall the linear program $LP(S')$ that defines $val(S')$. Let $\{f^*\}_{mn}$ be a set of optimal solution to $LP(S')$. We have by the definition of $\tilde{V}_{S'}$,

$$\sum_{m \in \tilde{V}_{S'}} \sum_{n \in S'} f^*_{mn} = \sum_{m \in \tilde{V}_{S'}} \sum_{n \in \tilde{S}'} f^*_{mn} \leq \sum_{n \in \tilde{S}'} \mu_n. \tag{3.15}$$

Furthermore,

$$\sum_{m \notin \tilde{V}_{S'}} \sum_{n \in S'} f^*_{mn} \leq \sum_{m \notin \tilde{V}_{S'}} \lambda^v_m \leq \sum_{m \in V_{S'}} \sum_{n \in S' \setminus \tilde{S}'} \lambda^*_{mn}, \tag{3.16}$$

where the first part follows from Constraint (3.3) of $LP(S')$ and the second part follows from (3.14). Hence, combining (3.15) and (3.16), we have,

$$\begin{aligned} val(S') - \sum_{n \in \tilde{S}'} \mu_n &= \sum_{m \in \tilde{V}_{S'}} \sum_{n \in S'} f^*_{mn} + \sum_{m \notin \tilde{V}_{S'}} \sum_{n \in S'} f^*_{mn} - \sum_{n \in \tilde{S}'} \mu_n \\ &\leq \sum_{m \notin \tilde{V}_{S'}} \sum_{n \in S'} f^*_{mn} \leq \sum_{m \in V_{S'}} \sum_{n \in S' \setminus \tilde{S}'} \lambda^*_{mn}, \end{aligned}$$

which conclude the proof of Claim 4.

Again, by Claim 3, we have

$$\begin{aligned}\sum_{l \in U} \sum_{n \in S' \backslash \tilde{S}'} \lambda^*_{ln} &\geq \sum_{l \in U_{S'}} \sum_{n \in S' \backslash \tilde{S}'} \lambda^*_{ln} \\ &= \sum_{l \in U_{S'}} \sum_{n \in S'} \lambda^*_{ln} = \sum_{l \in U_{S'}} \lambda^u_l. \end{aligned} \tag{3.17}$$

Then, summing up constraints (3.9) of $\mathcal{P}$ over $n \in S' \backslash \tilde{S}'$, we obtain

$$\sum_{n \in S' \backslash \tilde{S}'} r^*_n \geq \sum_{n \in S' \backslash \tilde{S}'} \sum_{l : n \in S_{u_l}} \lambda^*_{ln} + \sum_{n \in S' \backslash \tilde{S}'} \sum_{m : n \in S_{v_m}} \lambda^*_{mn} - \sum_{n \in S' \backslash \tilde{S}'} \mu_n \tag{3.18}$$

$$\geq \sum_{l \in U_{S'}} \sum_{n \in S' \backslash \tilde{S}'} \lambda^*_{ln} + \sum_{m \in V_{S'}} \sum_{n \in S' \backslash \tilde{S}'} \lambda^*_{mn} - \sum_{n \in S' \backslash \tilde{S}'} \mu_n \tag{3.19}$$

$$\geq \sum_{l \in U_{S'}} \lambda^u_l + val(S') - \sum_{n \in \tilde{S}'} \mu_n - \sum_{n \in S' \backslash \tilde{S}'} \mu_n \tag{3.20}$$

$$\geq \sum_{l \in U_{S'}} \lambda^u_l + val(S') - \sum_{n \in S'} \mu_n = val(S') - \Delta(S') > 0, \tag{3.21}$$

where Inequality (3.18) follows from rearrangement of the sums, Inequality (3.19) follows from (3.17) and Inequality (3.20) follows from Claim 4. Observe that by definition of $\tilde{S}'$, $\sum_{n \in S' \backslash \tilde{S}'} r^*_n = 0$, which contradicts (3.21). Hence, there must exists a u_l such that $S_{u_l} \in \tilde{S}'$. This concludes the proof of the lemma. □

Based on Lemma 8, invoking Proposition 6, we have that for all $n \in \tilde{S}'$, for any $\delta > 0$, $\lim_{t \to \infty} \mathbb{P}\{|\frac{Q_n(t)}{t} - r^*_n| < \delta\} = 1$, i.e., $\frac{Q_n(t)}{t}$ converges to r^*_n in probability. As in our system, $\lim_{t \to \infty} \frac{Q_n(t)}{t}$ exists almost surely, we have that $\frac{Q_n(t)}{t}$ converges to $r^*_n > 0$ with probability 1 for all $n \in \tilde{S}'$. Since there exists user dispatcher u_l with $S_{u_l} \subseteq \tilde{S}'$, it implies that $\lim_{t \to \infty} \frac{\sum_{n \in \tilde{S}'} Q^u_n(t)}{t} > \delta$ with probability 1 for some $\delta > 0$. As $Q^u_n(t)/t$'s are non-negative random variables, $\lim_{t \to \infty} \frac{\sum_{n \in \tilde{S}'} \mathbb{E}[Q^u_n(t)]}{t} > 0$. It follows that $\exists n \in \tilde{S}', \lim_{t \to \infty} \frac{\mathbb{E}[Q^u_n(t)]}{t} > 0$. □

Based on Theorem 6, we obtain the following corollary.

Corollary 4. *If the network DoS problem is feasible, then there exists a non-empty subset of user dispatchers U', such that if all the adversary dispatchers that have* *connection to $S_{U'} = \bigcup_{u_l \in U'} S_{u_l}$ send packets to $S_{U'}$ according to the JSQ rule, the system will become a JSQ server farm. Let $\mathbf{r}^*$ be the solution to the optimization problem $\mathcal{P}$ that corresponds to the JSQ server farm, then $\mathbf{r}_n^* > 0$ for all $n \in S_{U'}$.*

Proof. From Theorem 6, Lemma 8 and their proofs, it is straightforward to show that S_{u_l} in Lemma 8 is one that satisfies the condition of the corollary. □

Theorem 6 and Corollary 4 together imply that if the network DoS problem is feasible, then the adversary can achieve the goal by injecting traffic to the subset $S_{U'}$ of servers that correspond to some subset U' of user dispatchers following the JSQ rule. Furthermore, when the adversary dispatchers do so, all the queues in the aforementioned subset will grow with time. In the sequel, we will refer to such subsets of server as vulnerable sets and define the collection of vulnerable sets as $\mathcal{S}_0$. Formally, $\mathcal{S}_0 = \{S_{U'} \mid U'$ satisfies the condition in Corollary 4$\}$. Typically $\mathcal{S}_0$ is a sub-collection of all subsets of servers that satisfies the *val*-condition. For example, again consider Figure 3-2, although S_2 and S_3 both satisfy the *val*-condition, S_3 is a vulnerable set ($S_3 \in \mathcal{S}_0$) but S_2 is not, since the queue with $s_1 \in S_2$ does not grow with time if all the adversary dispatchers inject traffic following JSQ to S_2.

Now, we have discovered an optimal adversary injection policy that only requires the adversary dispatchers to perform JSQ routing to a vulnerable set of servers. Such JSQ-operation can automatically allocate budget to servers in some vulnerable set without "wasting" adversary traffic. To design an optimal oblivious policy, the remaining task is to identify a vulnerable set without relying on network statistics, which is what we will do in the following section.

3.4.2 The Min-Zero Policy

In this section, we present the optimal oblivious adversary injection policy – the Min-Zero policy. It uses the idea of the Target-JSQ policy and aims to identify a vulnerable set based (only) on queue-length information.

At each time slot t, the adversary maintains a target subset of user dispatchers and a corresponding target subset of servers, which are denoted as $U(t)$ and $S(t)$, with $U(t) \subseteq U$, $S(t) \subseteq S$ and $S(t) = \bigcup_{u_l \in U(t)} S_{u_l}$. All the adversary dispatchers that have connections to $S(t)$ send packets to $S(t)$ in a JSQ fashion, and other adversary dispatchers send packets arbitrarily. Then, after the servers finished their service at current slot, the adversary checks if $\min_{n \in S(t)} Q_n(t) = 0$ (hence the name, Min-Zero). If so, then at next slot, the adversary choose $U(t+1)$ uniformly at random from all non-empty subsets of user dispatchers and set $S(t+1)$ accordingly; otherwise, set $U(t+1) = U(t)$ and $S(t+1) = S(t)$. We formally present the Min-Zero policy in **Algorithm 4**.

Algorithm 4 The Min-Zero Policy

```
Input: Server set $S = \{s_1, \dots, s_n\}$, user dispatcher set $U = \{u_1, \dots, u_l\}$, adversary
    dispatcher set $V = \{v_1, \dots, v_m\}$
1: Initialize: $U(0) :=$ a random non-empty subset of $U$
    $S(0) := \bigcup_{u_l \in U(0)} S_{u_l}$.
2: for $t = 0, 1, 2, \dots$ do
3:     Each adversary dispatchers $v_m$ such that $S_{v_m} \cap S(t) \neq \emptyset$ sends packets to
       $S_{v_m} \cap S(t)$ following the JSQ rule.
4:     All other adversary dispatchers send packets arbitrarily. After the service phase
       of current time slot t
5:     if $\min_{n \in S(t)} Q_n(t) = 0$ then
6:         $U(t+1) :=$ nonempty subset of $U$ chosen uniformly at random.
7:         $S(t+1) := \bigcup_{u_l \in U(t+1)} S_{u_l}$
8:     else
9:         $U(t+1) := U(t), S(t+1) := S(t)$.
```

Obviously, the Min-Zero policy is oblivious. We now proceed to establish its optimality in Theorem 7.

Theorem 7. *Under the Min-Zero policy, there exists a queue n with $\lim_{t\to\infty} \frac{\mathbb{E}[Q_n^u(t)]}{t} > 0$ if the network DoS attack problem is feasible.*

Proof. First, by the execution of the Min-Zero Policy, it is straightforward to see that the evolution of the system follows an irreducible discrete-time countable-state Markov Chain with state $(Q(t), S(t))$. We will prove the theorem by invoking Lemma 7 with a suitably constructed Lyapunov function.

The Lyapunov function we define on the state space is

$$f(\mathbf{Q}(t), S(t)) = \mathbb{1}\{S(t) \in \mathcal{S}_0, \min_{n \in S(t)} \{Q_n(t)\} > TC_1\} \cdot \left(\min_{n \in S(t)} \{Q_n(t)\} - TC_1 \right),$$

with $\mathbb{1}\{\cdot\}$ denoting the indicator function, T being a large number that does not grow with time[6] and will be specified later, and $C_1 = (M + L)C$ being an upper bound on the sum of arrivals/service that a queue can receive. Due to the boundedness of arrivals and services, we can first easily verify that the Markov Chain satisfies the second condition in Lemma 7. We proceed to show that it also satisfies the first condition, i.e., it has positive expected drift under certain set. Specifically, we will prove that

$$\mathbb{E}\left[f((\boldsymbol{Q}(t+T), S(t+T)) - f((\boldsymbol{Q}(t), S(t)) \mid f(\boldsymbol{Q}(t), S(t)) > 0\right] \geq \epsilon, \quad \forall t.$$

Note that since f only takes integer value, the above corresponds to setting $k = T$, $\gamma = 1/2$ and A_γ accordingly in Lemma 7.

Before presenting the technical proof, we first give some intuition. Notice that the construction of the Lyapunov function f ensures that it takes positive value only in the states where the adversary is targeting some vulnerable subset, and will not change its target in the following T time slots. Conditioning on such states, the T-slot drift is roughly equal to the drift of the minimum queue length in the target subset $S(t)$. Since $S(t)$ is vulnerable, we can take T to be suitably large (but still a constant) and invoke Proposition 6 to show that the queue lengths in $S(t)$ all grow at a positive rate over the T time slots. This suggests that the Markov Chain satisfies the first condition in Lemma 7.

To make the intuition concrete, we first lower bound the drift. Conditioning on any t such that $f(\boldsymbol{Q}(t), S(t)) > 0$, without loss of generality and for notational convenience, we assume $t = 0$. It follows that the adversary's target will not change

[6]Our choice of T will depend on the network size and network statistics, but the Min-Zero policy does not.

over the next T time slots. Therefore, $S(T) = S(0) \in \mathcal{S}_0$. It follows that

$$\begin{aligned}
&f((\boldsymbol{Q}(T), S(T)) - f((\boldsymbol{Q}(0), S(0)) \\
=&\mathbb{1}\{S(T) \in \mathcal{S}_0, \min_{n\in S(T)}\{Q_n(T)\} > TC_1\} \cdot \left(\min_{n\in S(T)}\{Q_n(T)\} - TC_1\right) \\
&- \mathbb{1}\{S(0) \in \mathcal{S}_0, \min_{n\in S(0)}\{Q_n(0)\} > TC_1\} \cdot \left(\min_{n\in S(0)}\{Q_n(0)\} - TC_1\right) \\
=&\mathbb{1}\{\min_{n\in S(T)}\{Q_n(T)\} > TC_1\} \cdot \left(\min_{n\in S(T)}\{Q_n(T)\} - TC_1\right) \\
&- \left(\min_{n\in S(0)}\{Q_n(0)\} - TC_1\right) \\
\geq &\left(\min_{n\in S(0)}\{Q_n(T)\} - TC_1\right) - \left(\min_{n\in S(0)}\{Q_n(0)\} - TC_1\right) \\
= &\min_{n\in S(0)}\{Q_n(T)\} - \min_{n\in S(0)}\{Q_n(0)\}
\end{aligned}$$

Hence, we have

$$\begin{aligned}
&\mathbb{E}\left[f((Q(T), S(T)) - f((Q(0), S(0)) \mid f(Q(0), S(0)) > 0\right] \\
\geq&\mathbb{E}\left[\min_{n\in S(0)}\{Q_n(T)\} - \min_{n\in S(0)}\{Q_n(0)\} \mid f(Q(0), S(0)) > 0\right]. \qquad (3.22)
\end{aligned}$$

We proceed to establish that the RHS of (3.22) is positive. Let $Q^*(0) = \min_{n\in S(0)} Q_n(0)$ and $\tilde{\boldsymbol{Q}}(0) = (Q^*(0), \ldots, Q^*(0))$. Based on this, we define random vector $\tilde{\boldsymbol{Q}}(t)$ as the resulting queue length vector at t under the Min-Zero policy starting from state $(\tilde{\boldsymbol{Q}}(0), S(0))$ at 0. We couple $\boldsymbol{Q}(t)$ and $\tilde{\boldsymbol{Q}}(t)$ in the same probability space by equaling their corresponding sequences of arrivals (to the dispatchers) and services on each sample path. We further define $\boldsymbol{Q}^o(t)$ as $\tilde{\boldsymbol{Q}}(t) - \tilde{\boldsymbol{Q}}(0)$. It is clear that $\boldsymbol{Q}^o(t)$ has the same distribution as a random vector that is the result of performing the same sequence of routing actions (as in $\tilde{\boldsymbol{Q}}(t)$) starting from a all-zero queue length vector under the alternate dynamics for $0 \leq t \leq T$(i.e., all the realized services equal the offered services and queue lengths can be negative). By Proposition 9, we have that on each sample path, $\boldsymbol{Q}_n(t) \geq \tilde{\boldsymbol{Q}}_n(t) - N_1(L + M)C$ for all n and $t \leq T$. It follows that $Q_n(t) \geq Q_n^o(t) + Q^*(0) - N_1(L + M)C$ for all n and $t \leq T$.

Observe that for $0 \leq t \leq T$, $\boldsymbol{Q}^o(t)$ evolves as the queue length vector of a server farm employing the JSQ routing under the alternate dynamics. Consider the sub-server farm formed by servers in $S(0)$, user dispatchers and adversary dispatchers that have connection to $S(0)$. Since $S(0) \in \mathcal{S}_0$, i.e., it is a vulnerable set, by Corollary 4, the optimal solution $\boldsymbol{r}^*$ of its corresponding optimization problem $\mathcal{P}$ satisfies $\boldsymbol{r}^* > \mathbf{0}$. Then, by Proposition 8, the optimal solution $\tilde{\boldsymbol{r}}^*$ to its corresponding optimization problem $\mathcal{P}'$ under the alternate dynamics satisfies $\tilde{\boldsymbol{r}}^* = \boldsymbol{r}^* > \mathbf{0}$. Set $\epsilon = \epsilon_{S(0)}$ as $\frac{\min_{n\in S(0)} \tilde{r}_n^*}{2\min_{n\in S(0)} \tilde{r}_n^* + 4C_1} > 0$. We invoke Corollary 3, and obtain that if $T > T_{\epsilon_{S(0)}}$ (defined in Corollary 3), for all t with $T_{\epsilon_{S(0)}} \leq t \leq T$,

$$\mathbb{P}\left\{\forall n \in S(0), \frac{Q_n^o(t)}{t} \geq \frac{\tilde{r}_n^*}{2}\right\} \geq 1 - \epsilon_{S(0)}.$$

Therefore, for all $t \geq T_{\epsilon_{S(0)}}$, we have

$$\begin{aligned}
&\mathbb{P}\left\{\forall n \in S(0), \frac{Q_n^o(t)}{t} \geq \frac{\tilde{r}_n^*}{2}\right\} \geq 1 - \epsilon_{S(0)}\\
\Longrightarrow\; &\mathbb{P}\left\{\forall n \in S(0), Q_n^o(t) \geq \frac{\tilde{r}_n^* t}{2}\right\} \geq 1 - \epsilon_{S(0)}\\
\Longrightarrow\; &\mathbb{P}\left\{\forall n \in S(0), \tilde{Q}_n(t) \geq \frac{\tilde{r}_n^* t}{2} + Q^*(0)\right\} \geq 1 - \epsilon_{S(0)}\\
\Longrightarrow\; &\mathbb{P}\left\{\forall n \in S(0), Q_n(t) \geq \frac{\tilde{r}_n^* t}{2} + Q^*(0) - N_1 C_1\right\} \geq 1 - \epsilon_{S(0)}\\
\Longrightarrow\; &\mathbb{P}\left\{\min_{n\in S(0)} Q_n(t) \geq \frac{\tilde{r}_n^* t}{2} + Q^*(0) - N_1 C_1\right\} \geq 1 - \epsilon_{S(0)}.
\end{aligned}$$

On the other hand, we also have $\min_{n\in S(0)} Q_n(t) \geq Q^*(t) - C_1 t$ with probability 1. Now, we set the previously mentioned T as $\max_{S(0)\in\mathcal{S}_0} \max\{T_{\epsilon_{S(0)}}, \frac{8N_1C_1}{\min_{n\in S(0)} \tilde{r}_n^*}\}$, we have that

$$\begin{aligned}
&\mathbb{E}\left[\min_{n\in S(0)}\{Q_n(T) - \min_{n\in S(0)}\{Q_n(0)\} \mid f(Q(0), S(0)) > 0\right]\\
\geq&(1 - \epsilon_{S(0)}) \cdot \frac{\min_{n\in S(0)} \tilde{r}_n^* T}{2} - (1 - \epsilon_{S(0)}) N_1 C_1 - \epsilon_{S(0)} C_1 T\\
\geq&\frac{\min_{n\in S(0)} \tilde{r}_n^* T}{4} - N_1 C_1 \geq N_1 C_1 > 0.
\end{aligned}$$

This establishes that the Markov Chain is transient. More importantly, by Lemma

7, conditioning on any initial condition $(\boldsymbol{Q}(0), S(0))$ such that $f(\boldsymbol{Q}(0), S(0))>0$, the stopping time

$$\tau(\omega) = \inf\{t \geq 1 : f((Q(t,\omega), S(t,\omega))) = 0\}$$

satisfies that $\mathbb{P}\{\tau = \infty\} > 0$. It is straightforward to establish that there exist $\epsilon_1 > 0$ and t_1 such that the system will reach from any initial condition to some state with positive value of f in t_1 steps with probability at least ϵ_1. Combining the previous reasoning, we have that there exists T_0 and $\epsilon_0 > 0$, such that $\mathbb{P}\{f(\boldsymbol{Q}(t), S(t)) > 0,\ \forall t > T_0\} > \epsilon_0$, which implies that on those set of sample paths (with probability at least ϵ_0), after T_0, the adversary dispatchers injects traffic to a vulnerable subset of servers according to JSQ and never change the target. Since the number of vulnerable subset is finite, invoking Proposition 6, we have that there exists a vulnerable subset S' such that $U_{S'} \neq \emptyset$ and with probability at least $\epsilon_0' > 0$,

$$\forall n \in S',\ \lim_{t\to\infty} \frac{Q_n(t)}{t} > \delta > 0,$$

where δ is determined by the optimization problem $\mathcal{P}$ corresponding to the server farm where adversary dispatchers injecting to S' according to JSQ. It follows that, $\exists n \in S'$, with probability at least ϵ_0', $\lim_{t\to\infty} Q_n^u(t)/t > \delta'$, for some $\delta' > 0$. Therefore, we conclude that on the whole sample space,

$$\exists n, \lim_{t\to\infty} \frac{\mathbb{E}\left[Q_n^u(t)\right]}{t} > 0. \tag{3.23}$$

Hence the Min-Zero policy achieves the goal and we conclude the proof. □

Remark: (i) *Switching Threshold:* in **Algorithm 4** (line 5), we set the switching threshold as 0. It is clear from the proof that the policy remains optimal under any other positive constant switching threshold. Intuitively, the convergence time of the algorithm, i.e., the time it takes the adversary to identify a vulnerable set and does not switch target further, depends on the threshold. An over-low a threshold would force the adversary to switch prematurely while an over-high threshold would make the adversary switch too infrequently. (ii) *Distributed Implementation:* **Algorithm 4** describes the Min-Zero policy in a centralized fashion, but the policy can be easily implemented in a distributed way. In the distributed implementation, each adversary dispatcher maintains its own target subset of servers, sneds traffic to the target set according to the JSQ rule, and switch if the minimum queue length in the target subset hits zero. The optimality of the distributed implementation can be shown using similar method. This feature also makes the Min-Zero policy more attractive in practice. We will explore the aforementioned two aspects of the policy in the simulations.

3.5 Simulations

In this section, we evaluate the Min-Zero policy through simulations. We focus on studying how the parameters of the network and the policy influence the performance of the policy. Specifically, on the network side, we investigate the effects that network size and network load have on the convergence time of the Min-Zero policy, respectively; on the policy side, we investigate the impact of switching threshold and distributed implementation on the convergence time. In the following, we will first introduce the simulation environment, and then present the simulation results.

3.5.1 Simulation Setting

Network Data: We use two sets of server network data. In the first set, for each N (number of servers) in $\{100, 150, 200, \ldots, 500\}$, we generate 20 networks with N servers, $\lfloor N/4 \rfloor$ user dispatchers and $\lceil N/10 \rceil$ adversary dispatchers. Each user dispatcher is connected to $\lceil N/5 \rceil$ servers (selected uniformly at random). The service rate at each server is a binomial random variable $B(\mu, 1/2)$ with μ uniformly sampled from $\{20, \ldots, 50\}$. The arrival rate at each user dispatcher is a binomial random variable $B(\mu, 1/2)$ with μ uniformly sampled from $\{20, \ldots, 50\}$. Each adversary dispatcher has connections to $\lfloor N/5 \rfloor$ servers and injection rate as a binomial random variable. The connections and the mean injection rate (budget) are randomly assigned such that the overall network load ρ (total arrival/total service) of the network equals 0.75.[7] In the second set, for each network load ρ in $\{0.75, 0.80, 0.85, 0.90, 0.95\}$, we generate 20 networks with 200 servers, 50 user dispatchers and 20 adversary dispatchers. The service rates and user traffic arrival rates are sampled similarly as in the first set, while the connections and budgets of adversary dispatchers are randomly generated such that the network load equals ρ in expectation.

Variants of Min-Zero policy: In the simulation, we run the Min-Zero policy with switching threshold (0,10,50,200) to evaluate its performance dependence on the threshold. We also run the distributed version of Min-Zero with threshold 0 (described in the final remark of Section 3.4) to compare the performance of centralized and distributed implementations.

Performance Metric: We use the convergence time of the policy as our performance metric. Theoretically, the convergence time of the policy is defined as the time when the adversary dispatchers identify a vulnerable subset and never switch after that. For ease of computation, in centralized Min-Zero policies, we calculate the convergence time as the time slot that the last "switch" happens, and in the distributed version, we calculate it as the first time slot such that the number of user packets in a queue exceeds 1000. By examining the queue length trajectories in our simulations, we have confirmed that the policies do converge in all the instances and the computed convergence times well approximate the theoretical definition. Note that the results presented in each setting are averaged over 20 network instances.

3.5.2 Simulation Results

Network Size

We plot the results on the first set of data (varying network sizes) in **Figure 3-3**. A somewhat counter-intuitive observation is that the convergence times of all variants of Min-Zero decrease with the size of the network. One possible explanation is that among the networks we generated, the larger ones may have a larger portion of

[7]The procedure does not guarantee that the generated instances are feasible. But we found that all the instances we generated in the simulations were indeed feasible.

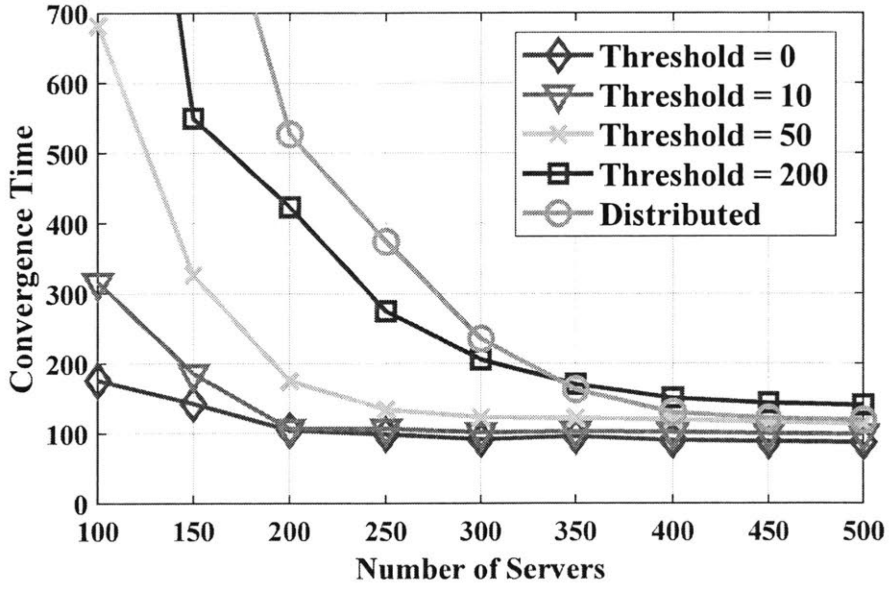


Figure 3-3: Convergence times of variants of Min-Zero on networks with different sizes.

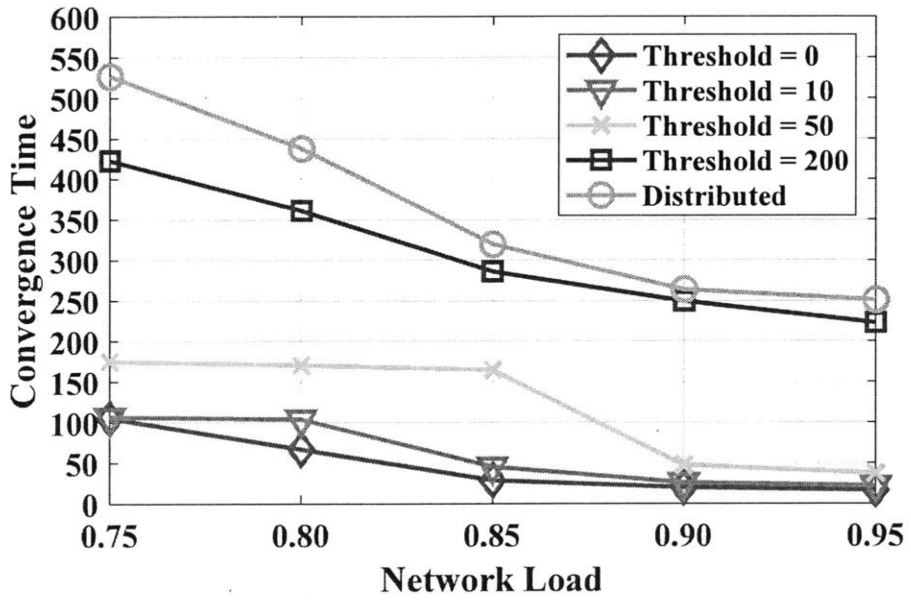


Figure 3-4: Convergence times of variants of Min-Zero on networks with different loads.

vulnerable subsets, making identifying a vulnerable subset easier. Another interesting finding is that, the higher the threshold, the more sensitive the policy is to the network size. This can be attributed to that in larger networks, the variance of arrival to servers is larger. Such variance directly affect the switching frequency of Min-Zero variants with larger threshold, which translates to influence on the convergence time.

Network Load

We plot the results on the second set of data (varying network loads) in Figure 3-4. We observe that the policies converge faster on networks with larger load. This is not surprising as increasing the load generally increases the number of vulnerable subsets, which in turn speeds up the convergence.

Variants of Min-Zero Policy

From **Figures 3-3 and 3-4**, we see that among the centralized policies, the convergence speed decreases as the switching threshold increases. This suggest that simply setting the threshold as zero may be the most desirable in practice. Furthermore, distributed version of Min-Zero converges slower than its centralized counterpart, but the gap becomes small in large networks.

3.6 Generalization to Multihop Networks

In this section, we extend our results to multi-hop networks. We first give the model and problem formulation in the multi-hop setting, and then present the counterparts of our previously obtained results in this setting. Since many of the proofs are similar to their single-hop counterparts and the notations are considerably heavier in the multi-hop case, we will only give proof sketches and focus the differences to the proof of single-hop results.

3.6.1 Model and Problem Formulation

Consider a network represented as directed graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$ with $\mathcal{N}$ denoting the set of nodes and $\mathcal{E} \in \mathcal{N} \times \mathcal{N}$ denoting the set of links. For each node $n \in \mathcal{N}$, let $Out(n)$ be its outgoing neighbors, i.e., $Out(n) = \{n' \in \mathcal{N}, (n, n') \in \mathcal{E}\}$. We assume that the network users send traffic from a single source $s \in \mathcal{N}$ to a single destination $d \in \mathcal{N}$.[8] To avoid unnecessary complexity, we require that there exists a path in $\mathcal{G}$ from each $n \in \mathcal{N} \backslash \{d\}$ to d. Additionally, there are a set of adversarial source nodes $\{v_1, \dots, v_M\}$ such that each node v_m in the set has connection to a subset $\mathcal{N}_{v_m} \subseteq \mathcal{N}$ of network nodes. Note that in our model, there are links from adversarial source nodes to network nodes, but no links in the opposite direction.

The system evolves in discrete time. Each network node $n \in \mathcal{N}$ is associated with a queue, with $Q_n(t)$ denoting the queue length at time t. For each link $e = (n, n') \in \mathcal{E}$, we use $b_e(t)$ or $b_{nn'}(t)$ to represent the offered transmission on the link at t, i.e., at most $b_e(t)$ packets can be sent through e. At each time slot, $\lambda_s(t)$ user packets arrive at source node s, and $\lambda_m^v(t)$ packets arrive at adversary source v_m, for $v_m \in \{v_1, \dots, v_M\}$. Similar as in the single-hop case, we assume that $b_e(t)$'s, $\lambda_s(t)$'s and $\lambda_m^v(t)$'s to be independent random variables with bounded support and are i.i.d across time with $\mathbb{E}[b_e(t)] = c_e, \mathbb{E}[\lambda_s(t)] = \lambda_s, \mathbb{E}[\lambda_m^v(t)] = \lambda_m^v$. Here, we can consider c_e's as the capacity of the links, λ_s as the user traffic rate, and $\boldsymbol{\lambda}^v = (\lambda_1^v, \dots, \lambda_M^v)$ as the adversary budget vector.

The network nodes (including s) employ the following routing policy. At time t, each node n except d performs the following procedure for each packet in its queue:

1. Pick n' with the minimum queue length $Q_{n'}(t)$ in $\{n' \in Out(n), Q_{n'}(t) > Q_n(t)$, number of packets transmitted from n to n' is less than $b_{nn'}(t)\}$.

2. If such n' exists, send the packet to n'; otherwise, hold the packet in the queue.

The destination node d instantly absorbs all the packets it receives, and thus $Q_d(t) = 0$ for all t. Such routing policy can be considered as each node sending packets to its

[8]Extension to multiple sources is straightforward by creating a super-source node.

outgoing neighbors according to the JSQ rule, which is a local implementation of the throughput-optimal back-pressure routing policy in our setting [16, 50]. Each adversarial source node injects traffic to the network nodes it has connection to following certain adversary policy. Without loss of generality, we assume that the capacities of links from adversarial source nodes to network nodes are infinity so that the amount of traffic that adversarial source nodes can inject is only constrained by the budget. Once sent to network nodes, the adversarial traffic will be merged with user traffic and delivered to d through the network nodes. Define $a^v_{mn}(t)$ as the packets injected from adversarial source v_m to node n with $\sum_{n\in\mathcal{N}_{v_m}} a^v_{mn}(t) = \lambda^v_m(t)$, $\tilde{b}_{nn'}(t)$ as the packets sent from n to n' at t, we can write the queue length evolution as:

$$\begin{aligned}Q_n(t+1) =& [Q_n(t) + \sum_{m:n\in\mathcal{N}_{v_m}} a^v_{mn}(t) + \mathbb{1}_{n=s}\lambda_s(t) + \\ & \sum_{n':n\in Out(n')} b_{n'n}(t) - \sum_{n'\in Out(n)} b_{nn'}(t)]^+.\end{aligned}$$

Similar as in the single-hop setting, we break down each Q_n into user traffic part Q^u_n and adversary traffic part Q^v_n. Here, we omit the evolution of Q^u_n and Q^v_m for ease of readability.

Based on the above preliminaries, we define the multihop version of network DoS attack problem.

Definition 6 (Multi-hop Network DoS Attack Problem). *The Multi-hop Network DoS Attack problem seeks an adversary injection policy under which there exists some network node n with* $\lim_{t\to\infty} \frac{\mathbb{E}[Q^u_n(t)]}{t} > 0$.

We give an example of the multi-hop network DoS attack problem is in Figure 3-5. In the example, the adversary can achieves the goal by v_1 injecting all traffic to n_1 and v_2 injecting all traffic to n_4.

3.6.2 Feasibility Region

Now, we proceed to present a necessary and sufficient condition for the Multi-hop Network DoS Attack problem to be feasible. It is based on a generalization of the

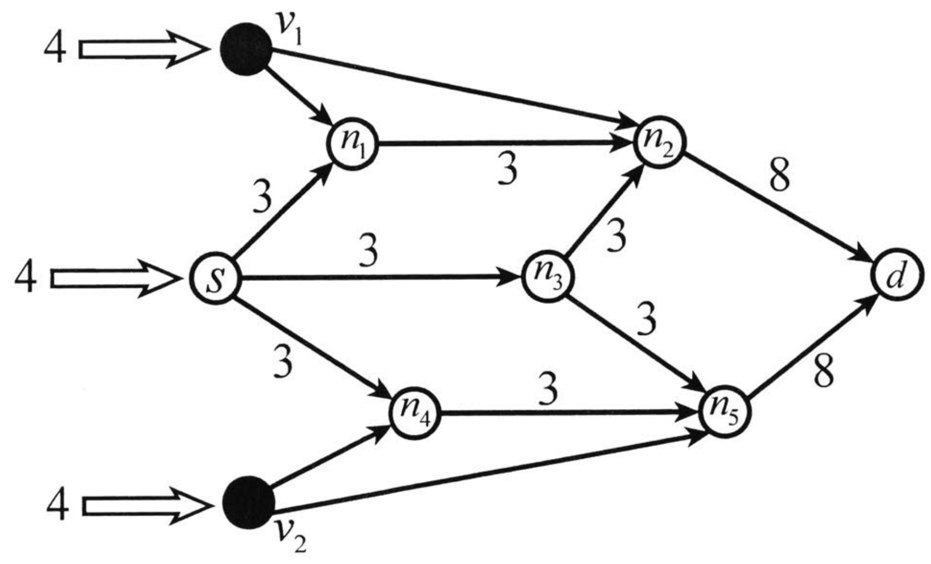


Figure 3-5: Illustration of the multi-hop network model. v_1 and v_2 are the adversary sources. The capacities are labeled beside the links.

val-condition.

We define s-d cut of the network as a partition $(\mathcal{S}, \mathcal{N}\backslash\mathcal{S})$ of network nodes such that $s \in \mathcal{S}$ and $d \in \mathcal{N}\backslash\mathcal{S}$. The capacity of cut $(\mathcal{S}, \mathcal{N}\backslash\mathcal{S})$ is defined as $Cap(\mathcal{S}) = \sum_{n\in\mathcal{S},n'\in\mathcal{N}\backslash\mathcal{S},(n,n')\in\mathcal{E}} c_{nn'}$. For each s-d cut $(\mathcal{S}, \mathcal{N}\backslash\mathcal{S})$, we further define $\Delta(\mathcal{S}) := Cap(\mathcal{S}) - \lambda_s$. We next generalize the *val* function to s-d cuts. Intuitively, for an s-d cut $(\mathcal{S}, \mathcal{N}\backslash\mathcal{S})$, $val(\mathcal{S})$ equals the maximum amount of traffic that the adversarial source nodes can send through $(\mathcal{S}, \mathcal{N}\backslash\mathcal{S})$. Formally, for $(\mathcal{S}, \mathcal{N}\backslash\mathcal{S})$, let $Out(\mathcal{S}) = \bigcup_{n\in\mathcal{S}} Out(n)$ and $V_{\mathcal{S}} = \{v_m \mid \mathcal{N}_{v_m} \cap \mathcal{S} \neq \emptyset\}$. We define a flow-network $\mathcal{G}_{\mathcal{S}}(\mathcal{N}_{\mathcal{S}}, \mathcal{E}_{\mathcal{S}})$ such that: $\mathcal{N}_{\mathcal{S}} = \mathcal{S} \cup Out(\mathcal{S}) \cup V_{\mathcal{S}} \cup \{s', d'\}$, i.e., $\mathcal{N}_{\mathcal{S}}$ consists of nodes in $\mathcal{S}$, $\mathcal{S}$'s outgoing neighbors, adversarial source nodes that have connection to $\mathcal{S}'$, a pseudo-source s' and a pseudo-destination d'; $\mathcal{E}_{\mathcal{S}}$ consists of the links between $\mathcal{S} \cup Out(\mathcal{S})$ in $\mathcal{E}$, the links from $V_{\mathcal{S}'}$ to $\mathcal{S}$, (s', n) for each $n \in V_{\mathcal{S}'}$ and (n, d') for each $n \in Out(\mathcal{S})$. The capacities of links in $\mathcal{E}_{\mathcal{S}}$ are defined as follows. For link (n, n') with $n, n' \in \mathcal{S} \cup Out(\mathcal{S})$, its capacity is equal to $c_{nn'}$. For link starting from the pseudo-source to adversarial source node (s', v_m), the capacity is equal to λ_m^v. The capacities of other links are infinity. Based on such capacitated flow network $\mathcal{G}_{\mathcal{S}}$, we define

$$val(\mathcal{S}) := \text{the maximum value of } s\text{-}d \text{ in } \mathcal{G}_{\mathcal{S}},$$

where the definition of *s-d* flow is standard [51] and we omit here. An illustration of the extended *val*-condition is given in Figure 3-6.

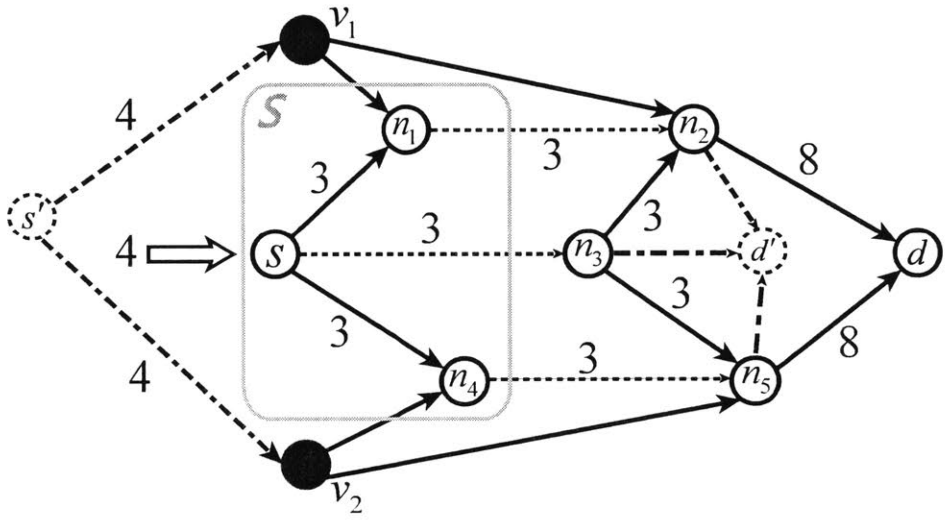


Figure 3-6: Illustration of the extended *val*-condition: Consider the *s-d* cut $(\mathcal{S}, \mathcal{N} \backslash \mathcal{S})$ with $\mathcal{S} = \{s, n_1, n_4\}$. $\lambda_s = 4$, $Cap(\mathcal{S}) = 9$ and $val(\mathcal{S}) = 6$. Therefore, the cut satisfies the extended *val*-condition.

Now, we are ready to define the extended *val*-condition, which leads to establishing the feasibility region of the multi-hop network DoS attack problem.

Definition 7 (Extended *val*-condition). *An s-d cut* $(\mathcal{S}, \mathcal{N} \backslash \mathcal{S})$ *satisfies the extended val-condition if* $val(\mathcal{S}) > \Delta(\mathcal{S})$.

Theorem 8. *The Multi-hop Network DoS Attack problem is feasible if and only if there exists an s-d cut that satisfies the extended val-condition.*

Proof. Note that under any adversary injection policy, the network $\mathcal{G}$ can be considered as a network that employs Back-pressure routing with multiple sources and single destination. The proof follows similar line as that of Theorem 5, and additionally relies on the following observation regarding single-commodity back-pressure network, which can be obtained from results in [50].

Observation 2. *For each sample path, if* $\lim_{t\to\infty} Q_n(t)/t > 0$ *for some* $n \in \mathcal{N}$*, then for all* $n' \in Out(n)$*, the time average traffic rate from* n *to* n' *equals* $c_{nn'}$*.*

We first show the sufficiency of the condition. If some cut $(\mathcal{S}, \mathcal{N}\backslash\mathcal{S})$ satisfies the extended *val*-condition, we consider the randomized adversary injection policy given by the max-flow that defines $val(\mathcal{S})$. Then on an arbitrary sample path (except a set of measure zero), there must exist node n with $\lim_{t\to\infty} Q_n(t)/t > 0$. Let $\mathcal{S}'$ be the set of nodes n with $\lim_{t\to\infty} Q_n(t)/t > 0$. We claim that $s \in \mathcal{S}'$, and that the adversary achieves the goal directly follows. To justify the claim, suppose for the sake of contradiction $s \notin \mathcal{S}'$, then the time average traffic rate from s to $\mathcal{S}'$ must be zero, and the overflow of $\mathcal{S}'$ is caused solely by adversary traffic. Let $\tilde{\mathcal{S}} = \mathcal{S}' \cap \mathcal{S}$. We have that the adversary traffic that goes through the cut $(\mathcal{S}, \mathcal{N}\backslash\mathcal{S})$ does not pass $\tilde{\mathcal{S}}$. Let $\tilde{\lambda}$ be the time average rate of adversarial traffic that goes from $\mathcal{S}\backslash\tilde{\mathcal{S}}$ to $\mathcal{N}\backslash\mathcal{S}$. By the definition of the *val* function, we have that

$$\sum_{n\in\mathcal{S}\backslash\tilde{\mathcal{S}}, n'\in\mathcal{N}\backslash\mathcal{S}} c_{nn'} - \tilde{\lambda} - \lambda \le Cap(\mathcal{S}) - val(\mathcal{S}) - \lambda < 0, \tag{3.24}$$

which leads to a contradiction since (3.24) implies that there exists node not in $\mathcal{S}'$ whose queue length also grows linearly with time at some positive rate. □

We now proceed to the necessity part. If there exists a policy that achieves the goal. There must exists a sample path at which the user part of some queue grows at positive rate with time. Pick such a sample path, let $\mathcal{S}$ be the set of nodes whose queue length grow with time at positive rate on the sample path. It is easy to see that $(\mathcal{S}, \mathcal{N}\backslash\mathcal{S})$ is an *s-d* cut. We proceed to show that it satisfies the extended *val*-condition. Let λ^v be the total adversarial traffic injection rate under the policy. Denote r^u, r^v respectively as the sum of user and adversary parts of queue length growth rates of nodes in $\mathcal{S}$, and p^u, p^v respectively as the total rates of user and adversary traffic that goes through the cut $(\mathcal{S}, \mathcal{N}\backslash\mathcal{S})$. Based on the definitions, we have $\lambda^v = p^v + r^v$ and $\lambda_s = p^u + r^u$. Further by Observation 2, we have $p^v + p^u = Cap(\mathcal{S})$. It follows that

$$\begin{aligned} val(\mathcal{S}) + \lambda_s &= p^u + r^u + val(\mathcal{S}) \\ &= p^u + val(\mathcal{S}) - Cap(\mathcal{S}) - r^u + Cap(\mathcal{S}) \\ &\geq p^u + p^v - Cap(\mathcal{S}) + r^u + Cap(\mathcal{S}) \\ &= r^u + Cap(\mathcal{S}) > Cap(\mathcal{S}). \end{aligned}$$

Hence, $(\mathcal{S}, \mathcal{N}\backslash\mathcal{S})$ satisfies the extended *val*-condition and we conclude the proof.

3.6.3 The Multi-hop Min-Zero Policy

Theorem 8 gives the feasibility region of the multi-hop network DoS attack problem based on the extended *val*-condition. In this section, we show that in the multi-hop case, there also exists an optimal adversarial injection policy that achieves the feasibility region without relying on the knowledge of network statistics. The policy we come up with, presented in **Algorithm 5**, is essentially a multi-hop version of the Min-Zero policy.

Algorithm 5 The Multi-hop Min-Zero Policy

```
Input: Multi-hop Network $\mathcal{G}(\mathcal{N}, \mathcal{E})$, adversary source nodes $V = \{v_1, \dots, v_m\}$
1: Initialize: $(\mathcal{S}(0), \mathcal{N}\backslash\mathcal{S}(0)) :=$ a random $s$-$d$ cut of $\mathcal{G}$.
2: for $t = 0, 1, 2, \dots$ do
3:     All adversary source nodes $v_m$ such that $\mathcal{N}_{v_m} \cap \mathcal{S}(t) \neq \emptyset$ send packets to $\mathcal{N}_{v_m} \cap \mathcal{S}(t)$ following the JSQ rule.
4:     All other adversary dispatchers send packets arbitrarily.
       After the transmission of current time slot $t$:
5:     if $\min_{n \in \mathcal{S}(t)} Q_n(t) = 0$ then
6:         $(\mathcal{S}(t+1), \mathcal{N}\backslash\mathcal{S}(t+1)) :=$ an $s$-$d$ cut chosen uniformly at random.
7:     else
8:         $\mathcal{S}(t+1) := \mathcal{S}(t)$.
```

We establish the optimality of the Multi-hop Min-Zero policy in the following theorem.

Theorem 9. *Under the Multi-hop Min-Zero policy, there exists a network node n with $\lim_{t\to\infty} \frac{\mathbb{E}[Q_n^u(t)]}{t} > 0$ if the multi-hop network DoS attack problem is feasible.*

Proof. (Sketch) The proof follows the same road-map of that of Theorem 7. First, we establish multi-hop counterparts of Propositions 6 and 7 in Proposition 10 which will be given in the Appendix 3.8.1. Following similar ideas and using Proposition 10, we can generalize Theorem 6 and Corollary 4 to multi-hop networks, that is, if the problem is feasible, then there exists an *s*-*d* cut that satisfies the extended *val*-condition and by each adversary source node v_m injecting to $\mathcal{N}_{v_m} \cap \mathcal{S}$ following the JSQ rule, all the queues in $\mathcal{S}$ grow with time, i.e., $\forall n \in \mathcal{S}$, $\lim_{t\to\infty} \frac{\mathbb{E}[Q_n(t)]}{t} > 0$. Define the set of all such *s*-*d* cuts as vulnerable cuts $\mathcal{S}_0$. Next, we again interpret the network dynamics under the Multi-hop Min-Zero policy as a Markov Chain with state $(\boldsymbol{Q}(t), \mathcal{S}(t))$. We construct the same Lyapunov function as in the proof of Theorem 7 by choosing an appropriate T:

$$
\begin{aligned}
f(\mathbf{Q}(t), S(t)) = &\mathbb{1}\{S(t) \in \mathcal{S}_0, \min_{n\in S(t)}\{Q_n(t)\} > TC_1\}\cdot \\
&\left(\min_{n\in S(t)}\{Q_n(t)\} - TC_1\right), .
\end{aligned}
$$

where $C_1 = (|\mathcal{N}| + M)C$ is an upper bound on the change of queue length at ann node in one time slot. By a similar coupling bound as Proposition 9, together with Proposition 10, we establish the positive expected T-slot drift of f conditioned on $f((\boldsymbol{Q}(t), \mathcal{S}(t)) > 0$. Invoking Lemma 7, we demonstrate that the Markov chain is transient. Finally, going through the same reasoning as in Theorem 7, we show that there exists n such that $\lim_{t\to\infty} \frac{\mathbb{E}[Q_n^u(t)]}{t} > 0$, which concludes the proof. □

3.7 Chapter Summary

In this chapter, we studied the fundamental limit of volume-based network DoS attack. Under single-hop server farm, we first proposed a necessary and sufficient condition on the budget of the adversary for a successful attack to be feasible, which served as the feasibility region of the network DoS attack. We then designed an optimal adversary injection policy that does not rely on knowledge of network statistics – Min-Zero, under which the attack is successful whenever the condition is met.

Complementing the theoretical analysis, we further evaluated the performance of the Min-Zero policy through simulations. Finally, we generalized our results including feasibility region of the attack and optimal injection policy to multi-hop networks.

3.8 Chapter Appendix

3.8.1 Proof of Propositions 6, 7 and Their Multi-hop Generalization

We prove Propositions 6 and 7 by applying the results in [47, 48]. As stating the results involve heavy additional notations and modeling in [47, 48], we do not present them here rigorously, but simply describe them at an intuitive level and show how to apply them to prove our propositions.

The relevant results are Theorem 1 in [47] and Corollary 4.4 in [48]. Theorem 1 in [47] considers the fluid model of switch networks that employ Max-Weight scheduling, which subsume the JSQ server farm considered in this paper as a special case. It establishes that the fluid model solution divided by time converges to the optimal solution of certain optimization problem. Projecting this onto our case, the optimization problem corresponds to $\mathcal{P}$ in Proposition 6 and $\mathcal{P}'$ in Proposition 7 under the alternative dynamics. Corollary 4.4 in [48] shows the convergence of scaled queue length vector of network (single-hop or multi-hop) under Max-Weight (Back-pressure) routing to the corresponding fluid model solution. Combining these two results, setting the parameter t (different with t in our model) in Corollary 4.4 of [48] as 1, it follows that in our model $\boldsymbol{Q}(t)/t$ converges to the solution to $\mathcal{P}$ (or $\mathcal{P}'$ under the alternate dynamics) in probability, which concludes the proof of Propositions 6 and 7.

One can straightforwardly extend Theorem 1 in [47] to the multi-hop network in our model. Combining again with Corollary 4.4 in [48], it leads to Proposition 10 that we use in extending our results to multi-hop networks. Consider a directed network $\mathcal{G}(\mathcal{N}, \mathcal{E})$ evolving in discrete time. The network $\mathcal{G}$ has a single destination node d and multiple sources. The external traffic arrivals and offered transmission of network links are independent random variables and are i.i.d. across time. The time average external traffic arrival rate at node n is denoted as λ_n ($\lambda_n = 0$ if n is not a source), and the time average offered transmission rate at link $(n, n') \in \mathcal{E}$ is denoted as $c_{nn'}$. Note that under our assumptions, the time average rates are equal to the mean of the corresponding random variables. The network operates under the (local) back-pressure policy introduced in Section 3.6. Consider the following optimization problem $\mathcal{P}_m$:

$$\min \sum_n r_n^2$$

$$\sum_{n':(n',n)\in\mathcal{E}} f_{n',n} + r_n + \lambda_n \geq \sum_{n':(n,n')\in\mathcal{E}} f_{nn'}, \qquad \forall n \in \mathcal{N}\backslash\{d\}$$

$$0 \leq f_{nn'} \leq c_{nn'}, \qquad \forall (n, n') \in \mathcal{E}$$

$$r_n \geq 0, \qquad \forall n \in \mathcal{N}.$$

It is easy to verify that $\mathcal{P}_m$ is a convex optimization problem and thus have a unique optimal solution. Now, we are ready to present the proposition in the multi-hop setting.

Proposition 10. *Let $\mathbf{r}^* = (r_1^*, \ldots, r_n^*)$ be the optimal solution to the optimization problem $\mathcal{P}_m$ associated with the multi-hop network. The queue lengths satisfy that for any $\delta > 0$*

$$\lim_{t\to\infty} \mathbb{P}\left\{ \left| \frac{Q_n(t)}{t} - r_n^* \right| < \delta \right\} = 1.$$

Note that Proposition 10 is the counterpart of Proposition 6 in the multi-hop setting. One can easily extend Proposition 7 to the multi-hop setting in almost identical fashion, which we omit here.

3.8.2 Proof of Proposition 9

Define $d_n(t)$ as $\tilde{Q}_n(t) - Q_n(t)$. We have $d_n(0) \leq 0$ for all n, and if we can show that $d_n(t) \leq N_1 LC$, then we prove the proposition. We first give three observations regarding $d_n(t)$:

1. $\sum_n d_n(t) = \sum_n d_n(0) \leq 0$ for all t.
2. Service at server n does not change $d_n(t)$.
3. If $d_n(t+1) > d_n(t) > 0$, then there exists an n' such that $d_{n'}(t) \geq d_n(t)$.

The three observations are justified as follows. Since in the alternate dynamics, $Q_n(t+1) = Q_n(t) + a_n(t) - b_n(t) = Q_n(t) + \sum_l a_{ln}(t) - b_n(t)$, we have

$$\begin{aligned}\sum_{n=1}^{N} Q_n(t) &= \sum_{n=1}^{N} Q_n(0) + \sum_{n=1}^{N}\sum_{l=1}^{L}\sum_{i=0}^{t-1} a_{ln}(i) - \sum_{n=1}^{N}\sum_{i=0}^{t-1} b_n(i) \\ &= \sum_{n=1}^{N} Q_n(0) + \sum_{l=1}^{L}\sum_{i=0}^{t-1} a_l(i) - - \sum_{n=1}^{N}\sum_{i=0}^{t-1} b_n(i) \end{aligned} \tag{3.25}$$

Since $\boldsymbol{Q}$ and $\tilde{\boldsymbol{Q}}$ evolve under the same sequence of $\{\boldsymbol{a}(i)\}$ and $\{\boldsymbol{b}(i)\}$, (3.25) validates the first observation. The second observation can be seen from the fact that after some service $b_n(t)$ for server n at time t, $Q_n(t)$ and $\tilde{Q}_n(t)$ both decrease by an amount of $b_n(t)$. Hence, $d_n(t)$ cannot be changed by services. The third observation can be established as follows: if at t, $d_n(t) > 0$ and $d_n(t)$ increases at the next time slot, then there must exist a dispatcher u_l (with non-zero arrival at t) such that following the JSQ rule, u_l sends packets to n under queue length vector is $\tilde{Q}_n(t)$ but does not send packets to n under $Q_n(t)$. It follows that there exists n' such that $\tilde{Q}_{n'}(t) \geq \tilde{Q}_n(t)$ while $Q_{n'}(t) \leq Q_n(t)$. This implies that $d_{n'}(t) = \tilde{Q}_{n'}(t) - Q_{n'}(t) \geq \tilde{Q}_n(t) - Q_n(t) = d_n(t)$.

By Observation (2), we do not need to consider services. Hence, we focus on arrivals to queues from dispatchers and proceed to prove the proposition. Suppose for the sake of contradiction that there exists a time t and server n_1 such that $d_{n_1}(t) > N_1 LC$. We take t_1 to be the smallest t that satisfies the above condition. Since at each time slot, the total arrival to a server is at most LC, $t_1 > N_1$. Due to the same reason, we have $d_n(t_1-1) > (N_1-1)LC$. By definition of t_1, we have $d_n(t) \le N_1 LC$ for $0 \le t \le t_1$, and in particular, $d_{n_1}(t_1-1) < d_{n_1}(t_1)$. Hence, by observation (3), there must exists another server n_2 such that $d_{n_2}(t_1-1) \ge d_{n_1}(t_1-1) > (N_1-1)LC$. Next, consider the function $d_{n_1}(t)+d_{n_2}(t)$, let t_2 be the largest time slot such that $0 \le t_2 < t_1-1$ and $d_{n_1}(t_2+1)+d_{n_2}(t_2+1) > d_{n_1}(t_2)+d_{n_2}(t_2)$. Such t_2 must exist as $d_{n_1}(0)+d_{n_2}(0) \le 0$ while $d_{n_1}(t_1-1)+d_{n_2}(t_1-1) > 2(N_1-1)LC$. Note that since $d_{n_1}(t)+d_{n_2}(t)$ does not increase from t_2+1 to t_1-1, we have for $t_2+1 \le t < t_1-1$

$$d_{n_1}(t)+d_{n_2}(t) > d_{n_1}(t_1-1)+d_{n_2}(t_1-1) > 2(N_1-1)LC.$$

Combining this with $d_n(t) \le N_1 LC$ for $n = n_1, n_2, t \le t_1$, we have $d_n(t_2+1) > (N_1-2)LC$ and $d_n(t_2) > (N_1-3)LC$ for $n = n_1, n_2$. Since $d_{n_1}+d_{n_2}$ increases at t_2, there must exists a dispatcher such that it sends packets to some $n \in \{n_1, n_2\}$ (w.l.o.g. $n = n_1$) under $\tilde{\boldsymbol{Q}}$ but does not send packets to either n_1 or n_2 under $\boldsymbol{Q}$. Following similar reasoning as in establishing observation (3), we have that there exists n_3 such that $d_{n_3}(t_2) \ge d_{n_1}(t_2) > (M-3)LC$. Applying the same argument with the largest time slot t_3 ($0 \le t_3 \le t_2-1$) at which the function $d_{n_1}(t)+d_{n_2}(t)+d_{n_3}(t)$ increases, we have for $t_3+1 \le t < t_2-1$

$$\begin{aligned} d_{n_1}(t)+d_{n_2}(t)+d_{n_3}(t) &> d_{n_1}(t_2-1)+d_{n_2}(t_2-1)+d_{n_3}(t_2-1) \\ &> 3(N_1-3)LC. \end{aligned}$$

And we have $d_n(t_3+1) > (3(N_1-3)-2N_1)LC = (N_1-9)LC$ and $d_n(t_3) > (N_1-10)LC$ for $n = n_1, n_2, n_3$. It follows that there exists another n_4 with $d_{n_4} > (N_1-10)LC$. Repeating such argument, we will find a time t_N such that $d_n(t_N) > LC$ for all n, which implies that $\sum_{n=1}^{N} d_n(t_N) > NLC$. This contradicts observation (1). Thus, we conclude the proof.

Bibliography

[1] S. T. Zargar, J. Joshi and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks", in *IEEE communications surveys & tutorials*, Vol. 15, No. 4, pp. 2046-2069, 2013

[2] G. S. Paschos and L. Tassiulas, "Sustainability of Service Provisioning Systems Under Stealth DoS Attacks", in *IEEE Trans. on Control of Network Systems*, Vol. 4, No. 4, pp. 749-760, 2017.

[3] M. Guirguis, A. Bestavros, I. Matta and Y. Zhang, "Reduction of quality (RoQ) attacks on internet end-systems". in *Proc. Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 2, pp. 1362-1372. IEEE, 2005.

[4] X. Luo and R. K. C. Chang, "On a New Class of Pulsing Denial-of-Service Attacks and the Defense", in *NDSS*, 2005.

[5] `https://www.msspalert.com/cybersecurity-research/kaspersky-lab-study-average-cost-of-enterprise-ddos-attack-totals-2m/`

[6] C. Kolias, G. Kambourakis, A. Stavrou and J. Voas, "DDoS in the IoT: Mirai and other botnets", in *Computer*, Vol. 50, No. 7, pp. 80-84, 2017.

[7] R. Braga, E. de Souza Mota and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow", in *IEEE LCN*, Vol. 10 pp. 408-415, 2010.

[8] A. Compagno, M. Conti, P. Gasti and G. Tsudik, "Poseidon: Mitigating interest flooding DDoS attacks in named data networking", in *IEEE LCN*, pp. 630-638, 2013.

[9] S. Neumayer, A. Efrat and E. Modiano, "Geographic max-flow and min-cut under a circular disk failure model", in *Computer Networks*, Vol. 77, pp. 117-127, 2015.

[10] S. Neumayer, G. Zussman, R. Cohen, and E. Modiano, "Assessing the vulnerability of the fiber infrastructure to disasters", in *IEEE/ACM Trans. on Networking*, Vol. 19, No. 6, pp. 1610-1623, 2011.

[11] A. Sanjab, S. Walid and T. Başar, "Prospect theory for enhanced cyber-physical security of drone delivery systems: A network interdiction game", *IEEE ICC*, 2017.

[12] R. K. Wood, "Deterministic network interdiction", in *Mathematical and Computer Modelling*, Vol. 17, No. 2, pp. 1-18, 1993.

[13] K. J. Cormican, D. P. Morton and R. K. Wood, "Stochastic network interdiction", in *Operations Research*, Vol. 46, No. 2, pp. 184-197, 1998.

[14] V. Gupta, M. H. Balter, K. Sigman and W. Whitt, "Analysis of join-the-shortest-queue routing for web server farms", in *Performance Evaluation*, Vol. 64, No. 9-12, pp. 1062-1081, 2007.

[15] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus and A. Greenberg, "Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services", in *Performance Evaluation*, Vol. 68, no. 11, pp. 1056-1071, 2011.

[16] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks", in *IEEE Conference on Decision and Control*, pp. 2130-2132, 1990.

[17] C. Phillips, "The network inhibition problem", in *Proc. ACM STOC*, 1993.

[18] R. Zenklusen, "Network flow interdiction on planar graphs", in *Discrete Applied Mathematics*, Vol. 158, No. 13, pp. 1441-1455, 2010.

[19] C. Burch, R. Carr, S. Krumke, M. Marathe, C. Phillips and E. Sundberg, "A decomposition-based pseudoapproximation algorithm for network flow inhibition", in *Network Interdiction and Stochastic Integer Programming*, pp. 51-68, 2003.

[20] J. O. Royset, and R. K. Wood, "Solving the bi-objective maximum-flow network-interdiction problem", in *INFORMS Journal on Computing*, Vol. 19, No. 2, pp. 175-184, 2007.

[21] C. Lim and J. C. Smith, "Algorithms for discrete and continuous multicommodity flow network interdiction problems", in *IIE Transactions*, Vol. 39, No. 1, pp. 15-26, 2007.

[22] D. Bertsimas, E. Nasrabadi and J. B. Orlin, "On the power of randomization in network interdiction", in *Operations Research Letters*, Vol. 44, No. 1, pp. 114-120, 2016.

[23] J. Zheng and D. A. Castañón, "Dynamic network interdiction games with imperfect information and deception", in *IEEE ICC*, pp. 7758-7763, 2012.

[24] J. C. Smith, M. Prince and J. Geunes, "Modern network interdiction problems and algorithms", in *Handbook of Combinatorial Optimization*, Springer, pp. 1949-1987, 2013.

[25] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan and D. P. Williamson, "Adversarial queuing theory" in *Journal of the ACM*, Vol. 48, No. 1, pp. 13-38, 2001.

[26] D. Gamarnik, "Stability of adaptive and nonadaptive packet routing policies in adversarial queueing networks" in *SIAM Journal on Computing*, Vol. 32, No. 2, pp. 371-385, 2003.

[27] A. Goel, "Stability of networks and protocols in the adversarial queueing model for packet routing", in *Networks: An International Journal*, Vol. 37, No. 4, pp.219-224, 2001.

[28] S. Lim, K. Jung and M. Andrews, "Stability of the max-weight protocol in adversarial wireless networks", in *IEEE/ACM Trans. on Networking*, Vol. 22, No. 6, pp. 1859-1872, 2014.

[29] Q. Liang and Modiano, "Network utility maximization in adversarial environments", in *IEEE INFOCOM*, pp. 594-602, 2018.

[30] Q. Liang and E. Modiano, "Minimizing Queue Length Regret Under Adversarial Network Models", in *Proc. of the ACM on Measurement and Analysis of Computing Systems*, Vol. 2, No. 1, pp.11, 2018.

[31] S. Wang and N. Shroff, "Security game with non-additive utilities and multiple attacker resources", in *Proc. of the ACM on Measurement and Analysis of Computing Systems*, Vol. 1, No. 1, pp.13, 2017

[32] M. H. Manshaei, Q. Zhu, T. Alpcan, T. BacÅ§ar and J-P Hubaux, "Game theory meets network security and privacy", in *ACM Computing Surveys*, Vol. 45, No. 3, pp. 25, 2013.

[33] D. Niyato and E. Hossain, "Dynamics of network selection in heterogeneous wireless networks: An evolutionary game approach", in *IEEE Trans. on Vehicular Technology*, Vol. 58, No. 4, pp. 2008-2017, 2009.

[34] I. Aad, JP. Hubaux and E. W. Knightly, "Impact of denial of service attacks on ad hoc networks", in *IEEE/ACM Trans. on Networking*, Vol. 16, No. 4, pp. 791-802, 2008.

[35] B. Wang, Y. Zheng, W. Lou and YT. Hou, "DDoS attack protection in the era of cloud computing and software-defined networking", in *Computer Networks*, Vol. 81 pp. 308-319, 2015.

[36] M. Ficco and M. Rak, "Stealthy denial of service strategy in cloud computing", in *IEEE Trans. on Cloud Computing*, Vol. 3, No. 1, pp. 80-94, 2015.

[37] A. Ben-Tal, L. El Ghaoui and A. Nemirovski, "Robust optimization", Vol. 28, Princeton University Press, 2009.

[38] C. Chekuri and M. Pal, "A recursive greedy algorithm for walks in directed graphs", in *IEEE FOCS*, 2005.

[39] V.V. Vazirani, "Approximation algorithms", Springer Science & Business Media, 2013.

[40] D. Granata and A. Sgalambro, "Network Interdiction through Length-Bounded Critical Disruption Paths: a Bi-Objective Approach", in *Electronic Notes in Discrete Mathematics*, Vol. 52, pp. 375-382, 2016.

[41] R.K. Ahuja, T. L. Magnanti and J. B. Orlin, "Network flows", Pearson Education, 2014.

[42] D. Bertsimas and JN. N. Tsitsiklis, "Introduction to linear optimization", Athena Scientific, 1997.

[43] M. Ripeanu, I. Foster and A. Iamnitchi, "Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design", in *IEEE Internet Computing Journal*, 2002.

[44] A. Björklund, T. Husfeldt and S. Khanna, "Approximating longest directed paths and cycles", in *International Colloquium on Automata, Languages, and Programming*, Springer, 2004.

[45] R.M. Karp, "Reducibility among combinatorial problems", in *Complexity of computer computations*, Springer, pp. 85-103, 1972.

[46] `https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/`

[47] D. Shah and D. Wischik, "Fluid models of congestion collapse in overloaded switched networks," in *Queueing Systems*, vol. 69, no. 2, pp: 121, 2011.

[48] D. Shah and D. Wischik, "Switched networks with maximum weight policies: Fluid approximation and multiplicative state space collapse," in *The Annals of Applied Probability*, vol. 22, no. 1, pp: 70-127, 2012.

[49] G. Fayolle, V. A. Malyshev and M. V. Men'shikov, "Topics in the constructive theory of countable Markov chains," Cambridge university press, 1995.

[50] L. Georgiadis, L. Tassiulas, "Optimal overload response in sensor networks", in *IEEE Trans. on Information Theory*, Vol. 52, No. 6, pp. 2684-2696, 2006.

[51] R. K. Ahuja, T. L. Magnanti and J. B. Orlin, "Network flows", 1988.