Robust Heuristics: Attacks and Defenses for Job Size Estimation in WSJF Systems

Erica Chiang, Nirav Atre, Hugo Sadok
Carnegie Mellon University

ACM Reference Format:
Erica Chiang, Nirav Atre, Hugo Sadok. 2022. Robust Heuristics: Attacks and Defenses for Job Size Estimation in WSJF Systems. In *ACM SIGCOMM 2022 Conference (SIGCOMM '22 Demos and Posters), August 22–26, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3546037.3546062

1 Introduction

Packet scheduling algorithms control the order in which a system serves network packets, which can have significant impact on system performance. Many systems rely on Shortest Job First (SJF), an important packet scheduling algorithm with many desirable properties. Classic results [3] show that SJF provably minimizes average job completion time, and recent work [1] shows that a variant of SJF also protects systems against algorithmic complexity attacks (ACAs), a particularly dangerous class of Denial-of-Service (DoS) attacks [4]. In an ACA, an adversary exploits the worst-case behavior of an algorithm in order to induce a large amount of work in the target system, causing a significant drop in goodput despite using only a small amount of attack bandwidth. SurgeProtector [1] demonstrated that using *Weighted* SJF (WSJF) – scheduling packets by the ratio of job size to packet size – significantly mitigates the impact of ACAs on *any* networked system.

There is just one problem: *how do we determine a packet's job size without running the job?* A common technique is to estimate job sizes using heuristics. In an adversarial setting, however, inaccuracies in job size estimation may be exploitable, re-opening the door to ACA vulnerabilities. In this work, we explore three strategies for using WSJF in practice and bound their vulnerability against ACAs. Our key findings are: (1) any heuristic that results in estimated job-size-to-packet-size ratios increasing monotonically with the true ratios will lead to perfect scheduling, thereby maintaining SurgeProtector's guarantees; (2) a heuristic that accurately separates jobs into job size categories can also protect a system against ACAs, but the guarantees are not as strong; and (3) preempting jobs that run for longer than their estimates does not guarantee bounds on an adversary's damage if the estimates are inaccurate.

2 Background and Motivation

Atre et al. [1] argue that in the absence of true job size information, we can use heuristics to estimate job sizes. In this context, a heuristic $\tilde{c}$ is a mapping from packets to estimated job sizes.[1] But in an adversarial setting, it is conceivable that incorrect estimates could undermine the guarantees of a system's protection against attacks. Besides heuristics, we also explore *preemption* (i.e., pausing a job and resuming it at a later point), another technique that may help protecting systems when job sizes are unknown.

2.1 Mathematical Framework

We first build a mathematical framework for analyzing the impact of adversarial traffic on a system. Each packet can be characterized by a packet size, $s(p)$ (the amount of data sent over the wire, in bits), and a job size, $c(p)$ (the time required to process the packet, in seconds). We define a packet's z-ratio as the ratio of its job size to packet size, noting that WSJF schedules packets by increasing z-ratio. Finally, we quantify the vulnerability of the system using the *Displacement Factor* (DF) [1], defined as the adversary's payoff relative to the amount of resources they invest into the attack:

$$DF = \frac{\text{Innocent traffic displaced (Gbps)}}{\text{Attack bandwidth used (Gbps)}}$$

2.2 WSJF and ACAs

In this section, we summarize the results of SurgeProtector [1] in the context of our heuristic-based approach to packet scheduling. SurgeProtector uses the DF to quantify the severity of an ACA, and shows that WSJF scheduling imposes an upper-bound of 1 on the DF. This implies that in order to displace 1 bps of innocent traffic, an adversary must invest *at least* 1 bps of their own bandwidth into the attack. Given the practical limitations of crafting and sending large volumes of data, a bounded DF greatly reduces the harm that an adversary can do to a system. In this paper, we aim to understand how these theoretical findings extend to practical settings where job sizes are not known *a priori*.

2.3 Incorrect Estimates

The accuracy of heuristics is crucial to maintaining DF guarantees. To illustrate why poorly designed heuristics can lead to an unbounded DF, we consider a heuristic that incorrectly estimates packets of a certain true job size, while all other packets are estimated correctly. Figure 1 demonstrates why incorrect estimates can be dangerous; in this example, all packets have unit size, such that WSJF orders packets by job size (represented by the packet width).

More formally, consider a heuristic that estimates the job size for adversarial packets as ϵ, allowing adversarial packets to have an arbitrarily small z-ratio as ϵ goes to 0. This implies that an attacker can push the system into overload using an infinitesimally small amount of their own bandwidth, displacing all innocent traffic in

[1] Here, we assume direct correspondence between true and estimated job sizes for simplicity. However, our analysis admits more sophisticated mappings (*e.g.*, probability distributions) as well.

SIGCOMM '22 Demos and Posters, August 22–26, 2022, Amsterdam, Netherlands

ACM ISBN 978-1-4503-9434-5/22/08.
https://doi.org/10.1145/3546037.3546062

the process and leading to an unbounded DF. Thus, incorrect job size estimates in a scheduling policy that relies on the job sizes of packets (*e.g.*, WSJF) can lead to arbitrarily bad DFs.

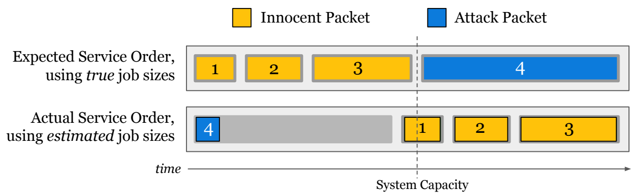

Figure 1: WSJF *should* de-prioritize the attack packet, but with incorrect estimates, innocent packets are displaced.

3 Novel Theoretical Findings

In this section, we present three novel theoretical findings regarding protection against ACAs when job sizes are unknown. Proofs for all theorems can be found in [2].

3.1 Strictly Monotonically Increasing Heuristics Maintain Perfect Scheduling

We first develop the concept of a 'perfect' heuristic, meaning that all packets are scheduled correctly when using estimated job sizes. Since correctly ordering all packets is equivalent to preserving the relative ordering between any pair of packets, a perfect heuristic must estimate job sizes such that between any two packets, the packet with smaller z-ratio will have a smaller *estimated* z-ratio. We can visualize this as any function mapping true ratios to estimated ratios that is *strictly monotonically increasing*, as seen in Figure 2.

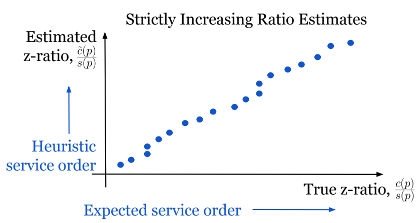

Figure 2: Strictly monotonically increasing ratios lead to perfect scheduling.

Any such heuristic preserves the relative ordering of packets as they are scheduled according to WSJF, which in turn maintains all guarantees from [1] and yields an upper-bound of 1 on the DF. In §A.1, we prove the following:

Theorem 1 (**DF of Monotonic Heuristic**). *Under WSJF, a heuristic $\tilde{c}$ is perfect if and only if $\frac{\tilde{c}(p)}{s(p)}$ is strictly monotonically increasing relative to $\frac{c(p)}{s(p)}$; such heuristics result in the DF being upper-bounded by* 1.

3.2 Step Functions Guarantee a Constant DF

In this section, we consider 'step function' heuristics in which packets are correctly classified into job size categories, but packets within each category ('step') are indistinguishable. In particular, we consider heuristics where the range of actual job sizes that each step covers has an upper bound that is a constant multiplicative factor, k, times the lower bound, and estimates of each step increase by the same factor k, as depicted in Figure 3.

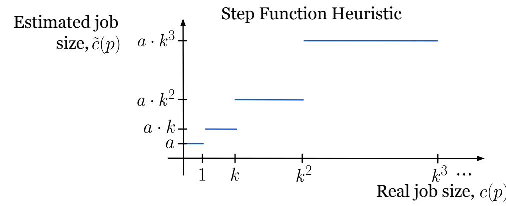

Figure 3: Step function heuristic.

Despite categorizing job sizes on a coarse level, the discrete steps still enforce a lower bound on how small each packet's job size estimate can be, protecting all jobs below a certain threshold. As we show in §A.2, this yields an upper-bound of k on the DF.

Theorem 2 (**DF of Step Function Heuristic**). *A heuristic of the form $\tilde{c}(p) = a \cdot k^{\lfloor \log_k c(p) \rfloor}$, where a is some arbitrary constant, results in the DF being upper-bounded by k.*

3.3 Preemption Can't Guarantee DF Bounds

Finally, we consider preemption as an additional aid to protecting systems against ACAs. The setup is as follows: each incoming job is assigned an estimated job size of J_p; if the job has not finished running within the allocated J_p time, the system preempts it and reinserts the job (with saved state) back into the scheduling queue, with an increased estimated job size of $2J_p$. The preemption model is depicted in Figure 4.

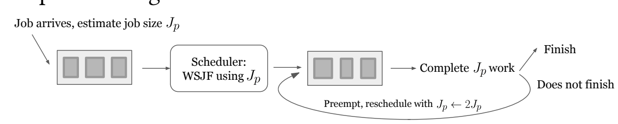

Figure 4: Preeemption system model.

This allows us to systematically allocate resources to each packet and ensures that packets finish according to job size order, even when job sizes are unknown. However, even if there is no preemption cost – an overly optimistic assumption – this setup can result in an unbounded DF. As we show in §A.3, preemption alone cannot guarantee any bound on the DF:

Theorem 3 (**DF of Preemptive Model**). *Under WSJF with preemption but without heuristics, there exist regimes of system parameters for which the DF is lower bounded by $\frac{\rho}{1-\rho}$, where $\rho \leq 1$ is the load on the system due to innocent traffic.*

4 Next Steps

Having identified desirable properties for heuristics and a framework for reasoning about their vulnerability, the main unanswered question is: how do we design data structures and corresponding heuristics such that we see these properties in practice? In addition, while we do not see theoretical bounds on the DF as a result of preemption alone, is it possible that some level of preemption could still be beneficial in practice?

Acknowledgments

We thank the anonymous reviewers for their insightful comments. This work was funded by Intel and VMware through the Intel/VMware Crossroads 3D-FPGA Academic Research Center, a VMWare Systems Research Award, and a Google Research Gift.

References

[1] Nirav Atre, Hugo Sadok, Erica Chiang, Weina Wang, and Justine Sherry. 2022. SurgeProtector: Mitigating Temporal Algorithmic Complexity Attacks using Adversarial Scheduling. In *Proceedings of the 2022 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*. Association for Computing Machinery, New York, NY, USA.

[2] Erica Chiang, Nirav Atre, and Hugo Sadok. 2022. *Robust Heuristics: Attacks and Defenses on Job Size Estimation for WSJF Systems.* Technical Report CMU-CS-22-117. Carnegie Mellon University, School of Computer Science. http://reports-archive.adm.cs.cmu.edu/anon/2022/CMU-CS-22-117.pdf

[3] Alan Cobham. 1954. Priority Assignment in Waiting Line Problems. *Journal of the Operations Research Society of America* 2, 1 (1954), 70–76.

[4] Scott A. Crosby and Dan S. Wallach. 2003. Denial of Service via Algorithmic Complexity Attacks. In *12th USENIX Security Symposium (USENIX Security 03)*. USENIX Association, Washington, D.C. https://www.usenix.org/conference/12th-usenix-security-symposium/denial-service-algorithmic-complexity-attacks