

Attack Resilience of Cache Replacement Policies

Tian Xie, Ting He, Patrick McDaniel, and Namitha Nambiar

Pennsylvania State University, University Park, PA, USA. Email: {tbx5027,tzh58,pdm12,nmn5265}@psu.edu

Abstract—Caches are pervasively used in computer networks to speed up access by reusing previous communications, where various replacement policies are used to manage the cached contents. The replacement policy of a cache plays a key role in its performance, and is thus extensively engineered to achieve a high hit ratio in benign environments. However, some studies showed that a policy with a higher hit ratio in benign environments may be more vulnerable to denial of service (DoS) attacks that intentionally send requests for unpopular contents. To understand the cache performance under such attacks, we analyze a suite of representative replacement policies under the framework of TTL approximation in how well they preserve the hit ratios for legitimate users, while incorporating the delay for the cache to obtain a missing content. We further develop a scheme to adapt the cache replacement policy based on the perceived level of attack. Our analysis and validation on real traces show that although no single policy is resilient to all the attack strategies, suitably adapting the replacement policy can notably improve the attack resilience of the cache.

Index Terms—cache replacement policy, access delay, DoS attack, attack resilience, TTL approximation.

I. INTRODUCTION

As one of the most widely-applied techniques in computer systems, caching can significantly boost system performance by storing and reusing previous computation or communication results. In the networking context, caches can serve requests close to the users, and thus reduce content access latency, network traffic load, and server workloads. Because of these benefits, they have been widely deployed in a variety of systems, e.g., World Wide Web (WWW) [1], [2], [3], Content Delivery Networks (CDNs) [4], Information Centric Networking (ICN) [5], and Domain Name System (DNS) [6]. In the emerging paradigm of Software Defined Networking (SDN), caches called flow tables are used to store controller instructions to alleviate the data-control plane bottleneck.

An attractive property of caches is that they are plug-and-play components that automatically adapt their contents to the current needs. At the core of this adaptation is a suite of replacement policies that decide which contents to evict to make room for new contents. There is a long series of works on developing and analyzing cache replacement policies, from simple First In First Out (FIFO) or Least Recently Used (LRU) to sophisticated policies involving virtual caches and

This work was supported by the National Science Foundation under award CNS-1946022. This research was also partly sponsored by the U.S. Army Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

multiple stages [7], [8]. However, most existing works only considered the performance in benign environments, where all the requests are from legitimate users.

Meanwhile, empirical studies in [9], [10] revealed that a policy with superior performance in benign environments can perform poorly under DoS attacks that flood the cache with requests for unpopular contents. For example, the Least Frequently Used (LFU) policy that is known to be optimal in the benign environment under the Independent Reference Model (IRM) [8] performs worse than LRU under attacks [9], [10], which is in turn worse than FIFO [10].

Motivated by these observations, we perform a comprehensive study of the attack resilience of cache replacement policies using the tool of Time-to-Live (TTL) approximations. These approximations not only allow us to explain how DoS attacks affect the hit ratios of the contents of interest, but also shed light on the optimal attack strategy and the defenses.

A. Related Work

Cache Replacement Policies: At a high level, cache replacement policies can be classified into *capacity-driven policies*, where a cached content is only evicted to make room for new content, and *TTL-based policies*, where a cached content is evicted after its TTL expires [11]. Traditionally, most policies are capacity-driven as they fully utilize the cache space, which can be further classified into recency-based policies (e.g., FIFO, LRU), frequency-based policies (e.g., LFU), randomized policies, and policies based on application-specific attributes (e.g., sizes, functions) [1], [2]. However, when maintaining consistency with the origin server is important, e.g., in DNS and WWW, TTL-based policies are popular [11]. In cases such as SDN, a combination of both types of policies is used [12]. The common objective of these policies is to maximize the cache hit ratio.

The performance of a single cache has been extensively studied. As exact analysis is difficult [7], various approximations have been developed, most notably the TTL approximation that models capacity-driven policies by TTL-based policies [13]. This idea has been used to predict the hit ratio for a number of capacity-driven policies, including FIFO, Random, LRU, and their variations [14], [8], [7]. The request processes under which these approximations apply have also been generalized from Poisson processes (i.e., IRM) [13] to renewal processes [8], Markov processes [7], and general stationary processes [15]. Besides known to be numerically accurate, TTL approximations are also shown to be asymptotically exact for large caches [16], [7], [15].

Application of cache replacement policies have also been studied in various systems, e.g., WWW [1], [2], [3], CDN [4], ICN [5], and DNS [6]. Many of these systems employ a network of interconnected caches, for which analytical results have been obtained under TTL approximations [6], [11], [17].

Most works on cache performance analysis assumed that a content is immediately available at the cache after a miss, which causes modeling error when *the cache has non-negligible content access delays*. This problem was first realized in [18], where new TTL approximations incorporating such delays were derived for FIFO, Random, and LRU. We will extend such analysis to a larger set of policies.

Attacks and Defenses: Caches have been the common targets of malicious attacks. In the networking context, caches can be used to extract private information [19], [20], [10], but the focus has been on degrading the cache performance by overwhelming its capacity [19], [21], [22], [23], [24] or occupying it with unpopular contents [19], [9], [25], [10], both effectively denying service to legitimate requests.

As for defenses, existing works mostly focused on using system-specific countermeasures to prevent/mitigate attacks (e.g., [26] for DNS, [27], [28] for ICN, [20], [21], [22], [23], [24] for SDN) or detecting attacks [29], [30], [9]. In contrast, we aim at understanding the *attack resilience of the cache itself*. Although attack resilience of caches has been briefly discussed in [31], [25], [31] only considered one attack strategy (similar to mice-flow attack considered in Section IV-A), and [25] only considered one replacement policy (FIFO), leaving open important questions such as: (i) How do popular replacement policies compare in terms of attack resilience? (ii) How does this comparison depend on the attack strategy? (iii) Is there a policy that is resilient to all the attack strategies? We will develop a tool (TTL approximation) to answer these questions analytically and provide explicit answers for representative policies and attack strategies.

B. Summary of Contributions

Our contributions are four-fold:

- 1) We extend the TTL approximation to incorporate the delays for the cache to obtain missing contents for a suite of state-of-the-art policies known to have superior performance in benign environments [8], which is of independent interest.
- 2) We use the obtained formulas to analyze the optimal attack strategy under a fixed total attack rate and its impact on the cache performance for legitimate requests.
- 3) Observing that the best policy in different attack scenarios can be different, we propose a scheme to adapt the replacement policy based on coarse parameters of the attack.
- 4) We perform a case study in SDN (flow table as cache) based on real traces. Besides confirming the accuracy of our analysis and the efficacy of the proposed policy adaptation scheme, our results also reveal relatively good resilience of two-staged policies, especially the one with FIFO eviction rule.

Roadmap. We will formulate our problem in Section II, present our TTL approximations in Section III, analyze the optimal attack strategy and its impact in Section IV, present our

policy selection scheme in Section V, present our experimental results in Section VI, and conclude the paper in Section VII.

II. PROBLEM FORMULATION

A. Request Arrival Model

Let F denote the set of all possible contents requested from the cache. Among these, a subset F_l contains the contents of interest to legitimate users, and its complement F_a contains the contents requested by the adversary during a DoS attack. We assume that $F_l \cap F_a = \emptyset$ as an intelligent adversary will never request anything of interest to legitimate users. We will use the Independent Reference Model (IRM) to obtain closed-form results, and discuss the generalization to arbitrary renewal processes when applicable.

Under IRM, the requests for each content $f \in F$ arrive according to an independent Poisson process with rate λ_f . Under the renewal model, the requests for each $f \in F$ arrive according to an independent renewal process with *inter-arrival distribution* $G_f(y)$, i.e., the i -th inter-arrival time Y_i satisfies $\Pr\{Y_i \leq y\} = G_f(y)$ for all $y \geq 0$. Let $\tilde{G}_f(y|t)$ denote the distribution function of the *excess life* at time t , i.e., if Z_t is the time from t to the next arrival, then $\Pr\{Z_t \leq y\} = \tilde{G}_f(y|t)$ for all $y \geq 0$. Let $m_f(t)$ denote the *renewal function*, defined as the expected number of arrivals in $(0, t]$. In the sequel, we will simply use “flow” to refer to a sequence of requests for the same content. Accordingly, we also refer to F as the set of all the incoming flows to the cache, F_l as the subset of legitimate flows, and F_a as the subset of attack flows.

B. Cache Model

Suppose that the cache under consideration has size C , measured in the number of distinct contents it can store. We adopt the common assumption that all contents are of equal size, as variable-sized contents can be split into equal-sized chunks for caching. When the cache is full, the cached contents are dynamically updated by its replacement policy. We consider a set of such policies as follows:

- **FIFO:** The *First In First Out (FIFO)* policy makes room for a new content by evicting the oldest cached content.
- **Random:** This policy evicts a randomly selected cached content to make room for a new content.
- **LRU:** The *Least Recently Used (LRU)* policy makes room for a new content by evicting the cached content that has not been requested for the longest time.
- **q -LRU:** This is a variation of LRU that only inserts a newly requested content into the cache with probability q .
- **LRU-2:** This is a two-staged policy that maintains a virtual cache (cache 1) storing content IDs and a real cache (cache 2) storing the actual contents, both employing the eviction rule of LRU. Each requested content ID not already in the virtual cache will be inserted into the virtual cache, but a requested content not already in the real cache will be inserted into the real cache if and only if its ID is already in the virtual cache. This policy can be extended to $k > 1$ caches, known as LRU- k , where caches $1, \dots, k-1$ are virtual caches and cache k is a real cache.

- *FIFO-2*: This is a two-staged policy similar to LRU-2, except that the eviction rule at each cache is FIFO.
- *Random-2*: This is another two-staged policy, except that the eviction rule at each cache is Random.

These policies can all be considered traffic-oblivious approximations to the *Least Frequently Used (LFU)* policy that statically stores the most popular contents, as LFU requires prior knowledge of content popularity. In a benign environment, LFU is known to have superior performance (optimal under IRM) [8], and some of the above policies can approximate LFU without requiring prior knowledge. Specifically, q -LRU tends to LFU as $q \rightarrow 0$, and LRU- k tends to LFU as $k \rightarrow \infty$, with much of the performance gain achieved at $k = 2$ [8].

While traditional cache performance analysis assumes that a content is immediately available at the cache after a miss¹, we consider a scenario more practical for network caches, where before inserting a missing content, the cache must first obtain the content from its origin server, which incurs a (possibly random) delay D referred to as the *access delay*. During the access delay, new requests of this content will incur misses but not generate further requests to the origin server. Let \bar{D} denote the mean access delay.

C. Objective

Our primary objective is to quantify the *attack resilience* of existing replacement policies in terms of how well they can preserve the hit ratios for legitimate users under DoS attacks, and develop new policies with better attack resilience. Our secondary objective is to improve the accuracy of TTL approximations by incorporating access delays.

III. TTL APPROXIMATION WITH ACCESS DELAY

Traditional TTL approximation formulas [8] are based on the assumption that the requested content is immediately available to the cache after a miss, which is too simplistic for network caches due to the access delay. It has been shown [18] that the existence of access delay causes notable deviation between the traditional TTL approximation and the actual hit ratio, where new TTL approximation formulas were developed to incorporate the impact of access delay for simple replacement policies including FIFO, Random, and LRU. Below, we will extend this study to a list of more sophisticated state-of-the-art replacement policies, by providing closed-form formulas under IRM (i.e., Poisson request arrivals) and generalizations under arbitrary renewal arrivals.

A. Review of Existing Results

In the presence of access delays, the following TTL approximations have been developed by [18]:

- *FIFO*: A FIFO cache can be modeled as a TTL cache with constant non-reset timers of timeout T [32]. Under Poisson arrivals, the hit ratio for content f is

$$h_f^{\text{FIFO}} = \frac{\lambda_f T}{1 + \lambda_f (\bar{D} + T)}. \quad (1)$$

¹Equivalently, each missing content will be available at the cache before the next request arrives.

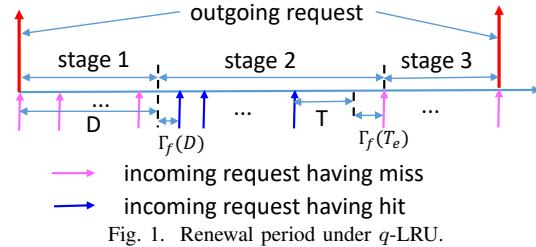


Fig. 1. Renewal period under q -LRU.

Under renewal arrivals, this hit ratio is

$$h_f^{\text{FIFO}} = \frac{\mathbb{E}[m_f(D+T)] - \mathbb{E}[m_f(D)]}{1 + \mathbb{E}[m_f(D+T)]}, \quad (2)$$

where the expectation is over the access delay D .

- *Random*: A Random cache can be modeled as a TTL cache with exponentially distributed non-reset timers with mean timeout \bar{T} [32]. Under Poisson arrivals, the hit ratio for content f is

$$h_f^{\text{Random}} = \frac{\lambda_f \bar{T}}{1 + \lambda_f (\bar{D} + \bar{T})}, \quad (3)$$

which is identical to (1) except that T is replaced by \bar{T} . Under renewal arrivals, this hit ratio is the same as (2), except that the expectation is over both D and T (which is exponentially distributed with mean \bar{T}).

- *LRU*: An LRU cache can be modeled as a TTL cache with constant reset timers of timeout T [32]. Under Poisson arrivals, the hit ratio for content f is

$$h_f^{\text{LRU}} = \frac{e^{\lambda_f T} - 1}{\lambda_f \bar{D} + e^{\lambda_f T}}. \quad (4)$$

Under renewal arrivals, this hit ratio is

$$h_f^{\text{LRU}} = \frac{\mathbb{E}[\tilde{G}_f(T|D)]}{(1 - G_f(T))(1 + \mathbb{E}[m_f(D)]) + \mathbb{E}[\tilde{G}_f(T|D)]}, \quad (5)$$

where the expectation is over D .

Here, the parameter T (or \bar{T}), known as the *characteristic time*, can be computed from the *characteristic equation*:

$$\sum_{f \in F} o_f = C, \quad (6)$$

where o_f is the cache occupancy probability of content f as a function of the characteristic time. Under Poisson arrivals, $o_f = h_f$ due to the PASTA property of Poisson processes.

B. TTL Approximation for q -LRU

The basic observation is that under renewal arrivals, the responses of the cache form renewal periods that are statistically identical to each other, and thus it suffices to analyze the hit ratio within a single renewal period. Below we will focus on a single content f as the analysis is identical for all contents.

As illustrated in Fig. 1, each renewal period starts when the cache forwards a request to the origin server and ends right before the next request that is forwarded to the origin server. Each period contains three stages: (i) stage 1 is the time D when the cache is waiting for the requested content from the origin server, during which all incoming requests will incur

misses, (ii) stage 2 is from the arrival of the content to (right before) the next miss, during which all incoming requests will incur hits, and (iii) stage 3 is from this miss to (right before) the next request from the cache to the origin server, during which all incoming requests will again incur misses. In the sequel, let X_i ($i = 1, 2, 3$) denote the number of incoming requests in stage i . The hit ratio for f is thus

$$h_f = \frac{\mathbb{E}[X_2]}{\mathbb{E}[X_1] + \mathbb{E}[X_2] + \mathbb{E}[X_3]}. \quad (7)$$

1) *Poisson Arrivals*: For stage 1, it is easy to see that $\mathbb{E}[X_1] = 1 + \lambda_f \bar{D}$, where the ‘1’ accounts for the arrival at the beginning of the period. For stage 2, since q -LRU behaves the same as LRU under hits and an LRU cache behaves like a TTL cache with reset timers and a constant timeout T [32], each new request generates a hit if and only if it arrives no later than T after the previous request, which occurs with probability $1 - e^{-\lambda_f T}$. Thus,

$$\Pr\{X_2 = n\} = (1 - e^{-\lambda_f T})^n e^{-\lambda_f T}, \quad n = 0, 1, \dots, \quad (8)$$

and $\mathbb{E}[X_2] = e^{\lambda_f T} - 1$. For stage 3, we know that by its definition, a q -LRU cache will only request the missing content from the origin server (to insert it into the cache) with probability q upon a miss, and thus the number of consecutive misses before the cache requests the content from the origin server is distributed as

$$\Pr\{X_3 = m\} = (1 - q)^m q, \quad m = 0, 1, \dots, \quad (9)$$

and $\mathbb{E}[X_3] = (1 - q)/q$. Plugging these results into (7) yields

$$h_f^{q\text{-LRU}} = \frac{e^{\lambda_f T} - 1}{\lambda_f \bar{D} + e^{\lambda_f T} + \frac{1-q}{q}}, \quad (10)$$

which reduces to (4) as $q \rightarrow 1$ as expected. The parameter T in (10) can be solved from $\sum_{f \in F} h_f^{q\text{-LRU}} = C$.

2) *Renewal Arrivals*: Without loss of generality, assume that $t = 0$ at the beginning of the renewal period under consideration. For stage 1, it is easy to see that $\mathbb{E}[X_1] = 1 + \mathbb{E}[m_f(D)]$, where the expectation is over D . For stage 2, each new request generates a hit if and only if it arrives no later than T after the timer resets, and the time between an arrival and the most recent timer reset is the excess life at D for the first arrival in stage 2 and an inter-arrival time thereafter. Thus,

$$\Pr\{X_2=n|D\}=\begin{cases} 1-\tilde{G}_f(T|D) & \text{if } n=0, \\ \tilde{G}_f(T|D)G_f(T)^{n-1}(1-G_f(T)) & \text{o.w.,} \end{cases} \quad (11)$$

and hence $\mathbb{E}[X_2] = \mathbb{E}[\tilde{G}_f(T|D)]/(1 - G_f(T))$, where the expectation is over D . For stage 3, we still have $\mathbb{E}[X_3] = (1 - q)/q$, as the number of consecutive misses before a q -LRU cache requests the content from the origin server (i.e., X_3) does not depend on the arrival process. Thus,

$$h_f^{q\text{-LRU}} = \frac{\frac{\mathbb{E}[\tilde{G}_f(T|D)]}{1-G_f(T)}}{1 + \mathbb{E}[m_f(D)] + \frac{\mathbb{E}[\tilde{G}_f(T|D)]}{1-G_f(T)} + \frac{1-q}{q}}, \quad (12)$$

where parameter T can be solved from (6). Detailed derivation of how to solve T is omitted due to space limitation.

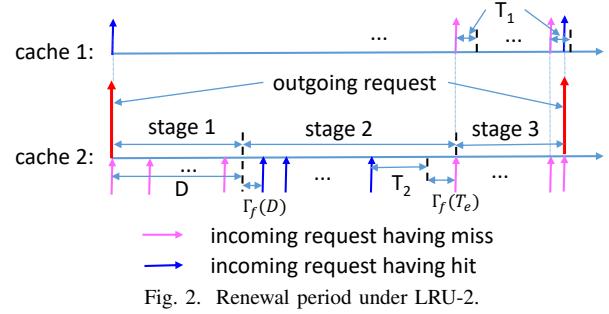


Fig. 2. Renewal period under LRU-2.

C. TTL Approximation for LRU-2

For a multi-staged policy such as LRU- k , the cache at each stage has its own renewal periods that are approximately independent across stages due to the vastly different characteristic times at different stages [8]. We thus need to analyze the renewal period at each stage. Below, we give detailed analysis for $k = 2$, but our approach extends to the general case.

As illustrated in Fig. 2, each renewal period of cache 2 (the real cache) is the time between consecutive requests to the origin server, and consists of three stages defined as in Section III-B. The difference is that an incoming request triggers an outgoing request to the origin server if and only if it results in a miss in cache 2 and a hit in cache 1 (the virtual cache).

1) *Poisson Arrivals*: The analysis for stages 1 and 2 remains the same as in Section III-B1, as the real cache behaves the same in these stages. That is, $\mathbb{E}[X_1] = 1 + \lambda_f \bar{D}$ and $\mathbb{E}[X_2] = e^{\lambda_f T_2} - 1$, where T_2 is the characteristic time of cache 2. For stage 3, we see by the definition of LRU-2 that X_3 is the number of consecutive misses in cache 1 before the next hit, which will trigger an outgoing request to the origin server and the starting of a new period. Since cache 1 is an LRU cache without access delay (as it only stores content IDs), we know from [32] that it behaves like a TTL cache with constant reset timers T_1 , which denotes its characteristic time. Moreover, as long as $T_2 \geq T_1$ (which holds when the two caches have the same size [8]), the first request in stage 3 must result in a miss in cache 1 because it arrives later than T_2 after the previous request (which is why stage 3 has started) and $T_2 \geq T_1$. Thus,

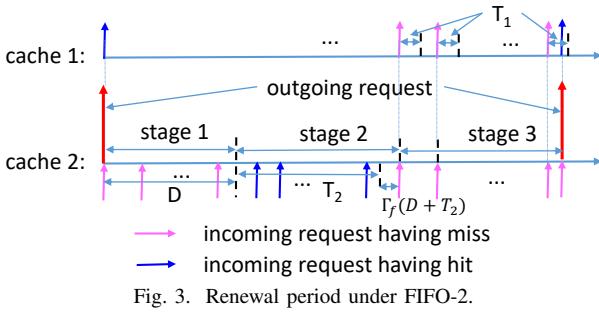
$$\Pr\{X_3-1=m\} = e^{-m\lambda_f T_1}(1 - e^{-\lambda_f T_1}), \quad m \geq 0, \quad (13)$$

and hence $\mathbb{E}[X_3] = 1/(1 - e^{-\lambda_f T_1})$. Plugging these results into (7) yields

$$h_f^{\text{LRU-2}} = \frac{e^{\lambda_f T_2} - 1}{\lambda_f \bar{D} + e^{\lambda_f T_2} + \frac{1}{1 - e^{-\lambda_f T_1}}}. \quad (14)$$

Here, T_1 is the solution to the characteristic equation of cache 1: $\sum_{f \in F} h_f^{\text{LRU}} = C_1$ (C_1 : size of cache 1), where there is no access delay, and T_2 is the solution to the characteristic equation of cache 2: $\sum_{f \in F} h_f^{\text{LRU-2}} = C$.

2) *Renewal Arrivals*: By similar arguments, we see from Section III-B2 that $\mathbb{E}[X_1] = 1 + \mathbb{E}[m_f(D)]$ for stage 1, and $\mathbb{E}[X_2] = \mathbb{E}[\tilde{G}_f(T_2|D)]/(1 - G_f(T_2))$ for stage 2, where both expectations are over D . For stage 3, the above analysis shows that $X_3 - 1$ is the number of consecutive inter-arrival times



in this stage that are greater than T_1 , the timeout value of the TTL approximation of cache 1. Thus,

$$\Pr\{X_3 - 1 = m\} = (1 - G_f(T_1))^m G_f(T_1), \quad m \geq 0, \quad (15)$$

and hence $\mathbb{E}[X_3] = 1/G_f(T_1)$. Plugging these results into (7) yields

$$h_f^{\text{LRU-2}} = \frac{\frac{\mathbb{E}[\tilde{G}_f(T_2|D)]}{1-G_f(T_2)}}{1 + \mathbb{E}[m_f(D)] + \frac{\mathbb{E}[\tilde{G}_f(T_2|D)]}{1-G_f(T_2)} + \frac{1}{G_f(T_1)}}, \quad (16)$$

where T_1 and T_2 can be computed from (6).

D. TTL Approximation for FIFO-2

Our analysis in Section III-C extends naturally to other multi-staged policies employing different eviction rules. Specifically, FIFO-2, as illustrated in Fig. 3, has renewal periods and three stages per renewal period that are defined in the same way as in Section III-C. The difference is that each cache follows the FIFO eviction rule.

1) *Poisson Arrivals*: For stage 1, we again have $\mathbb{E}[X_1] = 1 + \lambda_f \bar{D}$. For stage 2, as cache 2 behaves the same as a FIFO cache upon hits, which in turns behaves like a TTL cache with constant non-reset timers [32], this stage has a fixed duration T_2 (the characteristic time of cache 2), during which the expected number of incoming requests is $\mathbb{E}[X_2] = \lambda_f T_2$. For stage 3, again by the definition of two-staged policies, the number of requests X_3 in this stage is the number of consecutive misses in cache 1. Here, cache 1 is a FIFO cache without access delay, which behaves like a TTL cache with constant non-reset timers T_1 (the characteristic time of cache 1) [32]. Different from LRU-2, the first request in stage 3 may result in a hit in cache 1 (which triggers a request to the origin server and starts a new period), as there is no guaranteed gap between the last arrival in stage 2 and the first arrival in stage 3. Under the assumption that the two caches are independent (because T_2 is usually much larger than T_1) [8], this occurs with a probability equal to the hit ratio of cache 1, which is $\lambda_f T_1 / (1 + \lambda_f T_1)$ by (1). Conditioned on the first request in stage 3 incurring a miss in cache 1, each subsequent request incurs a miss in cache 1 if and only if the time between it and the previous request is greater than T_1 , which occurs with probability $e^{-\lambda_f T_1}$. Thus,

$$\Pr\{X_3 = m\} = \begin{cases} \frac{\lambda_f T_1}{1 + \lambda_f T_1} & \text{if } m = 0, \\ \frac{e^{-(m-1)\lambda_f T_1} (1 - e^{-\lambda_f T_1})}{1 + \lambda_f T_1} & \text{if } m > 0, \end{cases} \quad (17)$$

and hence $\mathbb{E}[X_3] = 1 / [(1 + \lambda_f T_1)(1 - e^{-\lambda_f T_1})]$. Plugging these results into (7) yields

$$h_f^{\text{FIFO-2}} = \frac{\lambda_f T_2}{1 + \lambda_f (\bar{D} + T_2) + \frac{1}{(1 + \lambda_f T_1)(1 - e^{-\lambda_f T_1})}}. \quad (18)$$

Here, T_1 is solvable from cache 1's characteristic equation: $\sum_{f \in F} h_f^{\text{FIFO}} = C_1$ (C_1 : size of cache 1), and T_2 is solvable from cache 2's characteristic equation: $\sum_{f \in F} h_f^{\text{FIFO-2}} = C$.

2) *Renewal Arrivals*: Under renewal arrivals, similar arguments show that $\mathbb{E}[X_1] = 1 + \mathbb{E}[m_f(D)]$ for stage 1, and $\mathbb{E}[X_2] = \mathbb{E}[m_f(D + T_2)] - \mathbb{E}[m_f(D)]$ for stage 2, both expectations over D . For stage 3, the arguments in Section III-D1 show that

$$\Pr\{X_3 = m\} = \begin{cases} \frac{m_f(T_1)}{1 + m_f(T_1)} & \text{if } m = 0, \\ \frac{1}{1 + m_f(T_1)} (1 - G_f(T_1))^{m-1} G_f(T_1) & \text{o.w.,} \end{cases} \quad (19)$$

where $m_f(T_1) / (1 + m_f(T_1))$ is the hit ratio of cache 1 obtained from (2) (where $D = 0$). Thus, $\mathbb{E}[X_3] = 1 / [(1 + m_f(T_1)) G_f(T_1)]$. Plugging these results into (7) yields

$$h_f^{\text{FIFO-2}} = \frac{\mathbb{E}[m_f(D + T_2)] - \mathbb{E}[m_f(D)]}{1 + \mathbb{E}[m_f(D + T_2)] + \frac{1}{(1 + m_f(T_1)) G_f(T_1)}}, \quad (20)$$

where T_1 and T_2 can be computed from (6).

E. TTL Approximation for Random-2

The analysis for Random-2 is very similar to that for FIFO-2, as both a FIFO cache and a Random cache behave like TTL caches with non-reset timers [32]. The difference, however, is that the timeout values for a Random cache are exponentially distributed (instead of being a constant as for FIFO), with a mean that equals the cache characteristic time. Specifically, each renewal period of Random-2 is still structured as in Fig. 3, except that T_2 and T_1 are exponential random variables², with means \bar{T}_2 and \bar{T}_1 that are the characteristic times of cache 2 and cache 1, respectively.

1) *Poisson Arrivals*: Similar to Section III-D1, we have $\mathbb{E}[X_1] = 1 + \lambda_f \bar{D}$, and $\mathbb{E}[X_2] = \lambda_f \bar{T}_2$. However, the analysis of X_3 is different. Under the independence assumption of the two caches [8], the first request in stage 3 results in a hit in cache 1 (i.e., $X_3 = 0$) with probability $\lambda_f \bar{T}_1 / (1 + \lambda_f \bar{T}_1)$, i.e., the hit ratio of cache 1 according to (3). Otherwise, each subsequent request results in a miss in cache 1 if and only if its inter-arrival time from the previous request is greater than the TTL of the content ID inserted into cache 1 by the previous request. Since the inter-arrival time and the TTL of cache 1 are both exponentially distributed with means $1/\lambda_f$ and \bar{T}_1 , respectively, the inter-arrival time is greater than the TTL with probability $1/(1 + \lambda_f \bar{T}_1)$. Thus,

$$\Pr\{X_3 = m\} = \left(\frac{1}{1 + \lambda_f \bar{T}_1} \right)^m \left(\frac{\lambda_f \bar{T}_1}{1 + \lambda_f \bar{T}_1} \right), \quad m \geq 0, \quad (21)$$

²More precisely, the TTL of each arrival into cache i ($i = 1, 2$) is an independent exponential random variable with mean \bar{T}_i .

which implies that $\mathbb{E}[X_3] = 1/(\lambda_f \bar{T}_1)$. Plugging these results into (7) yields

$$h_f^{\text{Random-2}} = \frac{\lambda_f \bar{T}_2}{1 + \lambda_f (\bar{D} + \bar{T}_2) + \frac{1}{\lambda_f \bar{T}_1}}, \quad (22)$$

where \bar{T}_1 is the solution to $\sum_{f \in F} h_f^{\text{Random}} = C_1$ (C_1 : size of cache 1), and \bar{T}_2 is the solution to $\sum_{f \in F} h_f^{\text{Random-2}} = C$.

2) *Renewal Arrivals*: Similar to Section III-D2, we have $\mathbb{E}[X_1] = 1 + \mathbb{E}[m_f(D)]$ for stage 1, and $\mathbb{E}[X_2] = \mathbb{E}[m_f(D + T_2)] - \mathbb{E}[m_f(D)]$ for stage 2, except that the second expectation is over both D and T_2 . For stage 3, the arguments in Section III-E1 show that $X_3 = 0$ with probability $\mathbb{E}[m_f(T_1)]/(1 + \mathbb{E}[m_f(T_1)])$ (expectation over T_1), which is the hit ratio of cache 1. Otherwise, for $m = 1, 2, \dots$,

$$\Pr\{X_3=m\} = \frac{\mathbb{E}[G_f(T_1)]}{1 + \mathbb{E}[m_f(T_1)]} (1 - \mathbb{E}[G_f(T_1)])^{m-1}, \quad (23)$$

where $1 - \mathbb{E}[G_f(T_1)]$ (expectation over T_1) is the probability for an inter-arrival time to be greater than the TTL of cache 1. Thus, $\mathbb{E}[X_3] = 1/((1 + \mathbb{E}[m_f(T_1)])\mathbb{E}[G_f(T_1)])$. Plugging these results into (7) yields

$$h_f^{\text{Random-2}} = \frac{\mathbb{E}[m_f(D + T_2)] - \mathbb{E}[m_f(D)]}{1 + \mathbb{E}[m_f(D + T_2)] + \frac{1}{(1 + \mathbb{E}[m_f(T_1)])\mathbb{E}[G_f(T_1)}}}. \quad (24)$$

Here, the only unknown parameters are the means \bar{T}_i of T_i for $i = 1$ and 2, which can be computed from (6).

Remark: Although seemingly similar, (20) and (24) differ subtly in that T_i ($i = 1, 2$) is treated as a constant in (20) but a random variable in (24), which implies that generally FIFO-2 and Random-2 perform differently in terms of hit ratio. They perform differently even under IRM, as seen from (18) and (22). This result is in contrast to the previous result that FIFO and Random have the same hit ratio under IRM [8].

IV. PERFORMANCE UNDER DOS ATTACK

We now apply the TTL approximation formulas obtained in Section III to analyze the performance of these policies under DoS attacks. Assuming that the cache cannot distinguish requests sent by the attacker from those sent by legitimate users (otherwise it can simply filter out requests from the attacker), we measure the performance of a given replacement policy under attack by its average hit ratio for the legitimate users. Under the TTL approximation, the hit ratios of different contents are only related through the characteristic time of the cache. Therefore, *attack flows affect the hit ratios of legitimate flows by affecting the characteristic time*.

Specifically, let $h^\pi(\lambda_f, T)$ denote the TTL approximation of the hit ratio of a content with request rate λ_f at a cache with policy π and characteristic time T . Given legitimate flows of individual rates $(\lambda_f)_{f \in F_l}$ and a total rate $\Lambda_l := \sum_{f \in F_l} \lambda_f$, we measure the performance of π by

$$\sum_{f \in F_l} \frac{\lambda_f h^\pi(\lambda_f, T)}{\Lambda_l}, \quad (25)$$

where T implicitly depends on the attack rates $(\lambda_f)_{f \in F_a}$ in addition to the legitimate flow rates $(\lambda_f)_{f \in F_l}$. We will focus on IRM in the rest of this section for explicit insights, although our approach is extensible to more general cases.

A. Optimal Attack Strategy

To understand the fundamental performance limit under DoS attacks, we first identify the optimal attack strategy against each policy. In particular, while a higher attack rate will always bring more damage, it also incurs a higher cost to the attacker. Therefore, the optimal attack strategy should make the best use of a given total attack rate.

1) *Attack Rate Allocation*: Given the total attack rate Λ_a , let $\mathcal{A} \subseteq \{(\lambda_f)_{f \in F_a} : \lambda_f \geq 0, \sum_{f \in F_a} \lambda_f = \Lambda_a, |F_a| = C_a\}$ be the set of candidate rate allocations among C_a attack flows.

Theorem IV.1. If the characteristic time T is constant for all $(\lambda_f)_{f \in F_a} \in \mathcal{A}$, and $h^\pi(\lambda_f, T)$ is an increasing function of T and a concave function of λ_f , then the optimal attack strategy in \mathcal{A} that minimizes (25) is to equally allocate the total attack rate, i.e., $\lambda_f = \Lambda_a/C_a$ for all $f \in F_a$.

Proof. Since the hit ratio of each legitimate content is increasing in T , the most effective attack strategy should minimize T . From the characteristic equation:

$$\sum_{f \in F_l} h^\pi(\lambda_f, T) + \sum_{f \in F_a} h^\pi(\lambda_f, T) = C, \quad (26)$$

we know that minimizing T , which will minimize the first term on the left-hand side of (26), is equivalent to maximizing the second term. This leads to a constrained optimization problem:

$$\max \sum_{f \in F_a} h^\pi(\lambda_f, T) \quad (27a)$$

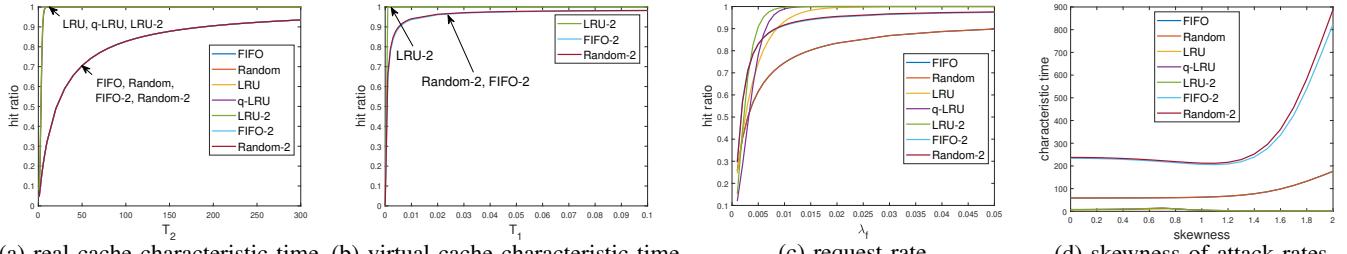
$$\text{s.t. } (\lambda_f)_{f \in F_a} \in \mathcal{A}. \quad (27b)$$

Since T can be treated as a constant and $h^\pi(\lambda_f, T)$ is concave in λ_f , we have

$$\frac{1}{C_a} \sum_{f \in F_a} h^\pi(\lambda_f, T) \leq h^\pi\left(\frac{\sum_{f \in F_a} \lambda_f}{C_a}, T\right) = h^\pi\left(\frac{\Lambda_a}{C_a}, T\right), \quad (28)$$

achieved when $\lambda_f = \Lambda_a/C_a$ for all $f \in F_a$. \square

We have verified that under IRM, all the policies considered in Section III satisfy the conditions in Theorem IV.1. Specifically, it is easy to verify that all the hit ratios are increasing in the characteristic time of the (real) cache. For the two-staged policies (LRU-2, FIFO-2, and Random-2), the hit ratio is also increasing in the characteristic time of the virtual cache. Moreover, for FIFO and Random, it is also easy to verify by checking the sign of the second derivative that the hit ratios are concave in the request rate. The same holds for LRU and q -LRU if \bar{D} is sufficiently small and $e^{\lambda_f T} \geq (1 - q)/q$. For the two-staged policies, analytical analysis of the concavity of the hit ratio as a function of the request rate becomes intractable, but we have verified numerically that the hit ratios are also concave in the request rate. Finally, while

Fig. 4. Verifying conditions of Theorem IV.1 (parameters of legitimate flows as in Section VI-A1, $\Lambda_a = 1000$, $C_a = 1000$).

the characteristic time will change under highly skewed rate allocations, it remains largely constant for a wide range of skewness parameters that correspond to potentially good attack strategies. See illustrative plots in Fig. 4 (similar observations hold under other parameter settings).

2) *Optimal #Attack Flows*: Under a fixed total attack rate and equal rate allocation, the attack strategy is fully determined by the number of attack flows C_a . In theory, we can plug the rates of legitimate and attack flows into the TTL approximation formulas to write (25) as a function of C_a , which can then be minimized to choose the optimal C_a . However, we see from plugging the hit ratio formulas into the characteristic equation (6) that the characteristic time T is the solution to a high-order polynomial or transcendental equation that cannot be solved in closed form, and thus (25) cannot be written as a closed-form function of C_a . Instead, we use other means to obtain insights, starting with the following observation.

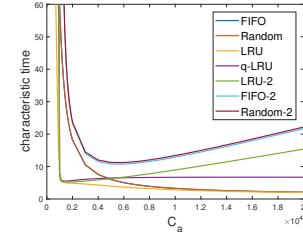
Proposition IV.2. Under FIFO, Random, and LRU, $C_a = \infty$ is optimal in minimizing (25) under IRM.

Proof. We prove the statement by arguing that $C_a = \infty$ minimizes the characteristic time T . Since FIFO and Random have the same hit ratio under IRM [8], it suffices to show the above for FIFO and LRU. To this end, we note that T represents the TTL of a newly inserted content f , which is the time for the cache to receive requests for C distinct contents other than f (assumed to be a constant independent of f under the TTL approximations) [32], [8]. Clearly, under a fixed total attack rate, associating every attack request with a distinct content minimizes the TTL and hence (25). \square

For the more advanced policies that perform selective insertion upon misses, we resort to numerical analysis. Specifically, as we have verified that the hit ratio is an increasing function of the characteristic time, it suffices to examine what value of C_a will minimize the characteristic time under each policy. Results under a sample parameter setting is shown in Fig. 5, but similar observations hold under other settings.

These results imply the following attack strategies:

- 1) *Mice-flow attack*, which sends as many attack flows as possible, each with a small rate, is most effective under the policies covered by Proposition IV.2.
- 2) *Elephant-flow attack*, which sends fewer attack flows such that each of them has a sufficiently high rate (relative to the legitimate flows), is most effective under policies with

Fig. 5. Optimizing #attack flows (parameters in Section VI-A1, $\Lambda_a = 1000$). selective insertion and discriminate eviction rules (e.g., q-LRU, LRU-2).

- 3) *Medium-flow attack*, with an intermediate number of attack flows, is most effective under policies with selective insertion but indiscriminate eviction rules (e.g., FIFO-2, Random-2).

For example, under the setting in Fig. 5, the optimal C_a for q-LRU and LRU-2 is around 1000, which makes the rates of attack flows comparable to that of the largest legitimate flow, hence suggesting an elephant-flow attack; the optimal C_a for FIFO/Random-2 is around 6000, leading to much smaller attack flows (smaller than the top 6 legitimate flows), suggesting a medium-flow attack; under FIFO, Random, and LRU, the attack becomes more effective as C_a increases, as predicted by Proposition IV.2, suggesting a mice-flow attack.

Remark: Although we use the average hit ratio as the performance metric, the optimal attack strategy will remain the same even if the adversary only targets at a specific flow or a subset of flows, as the hit ratio of every flow is increasing in the characteristic time.

B. Impact on Cache Performance

Given the optimal attack strategies, we can now plug them into the TTL approximation formulas of various policies to analyze the impact of the attacks on the hit ratios for legitimate users. Below, we only show the predicted hit ratio according to (25); validation based on actual hit ratios will be presented later in Section VI.

As the optimal attack will allocate equal rate to all the attack flows, it suffices to parameterize an attack by the total attack rate Λ_a and the number of attack flows C_a . We start by confirming our previous observations regarding the optimal design of C_a under a fixed total attack rate $\lambda_a C_a$ in Fig. 6 (a). As in Fig. 5, we see that the policies divide into three groups: (i) FIFO, Random, and LRU are most vulnerable to mice-flow attacks corresponding to large C_a , (ii) q-LRU and LRU-2

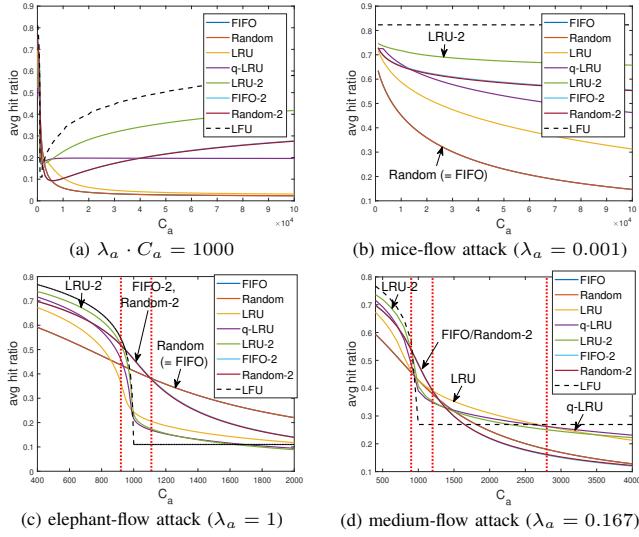


Fig. 6. Predicted performance under DoS attack (parameters of legitimate flows as in Section VI-A1, under which $\lambda_f \in [0.0002, 1]$ for $f \in F_i$).

are most vulnerable to elephant-flow attacks corresponding to relatively small C_a (≈ 1000), and (iii) FIFO-2 and Random-2 are most vulnerable to medium-flow attacks corresponding to an intermediate C_a (≈ 6000).

While the behaviors of FIFO, Random, and LRU have been explained by Proposition IV.2, the behaviors of the other policies also have intuitive explanations. Specifically, we know that q -LRU and LRU-2 closely approximate LFU [8], which only serves the largest C flows. Hence, these policies will effectively preserve the hit ratio for legitimate users if the largest C flows do not include attack flows, but severely degrade this value as more and more of the largest C flows become attack flows. To illustrate this point, we fix the rate per attack flow at λ_a and vary the number of attack flows C_a , as shown in Fig. 6 (b–d). We have also added the curve for LFU. The results confirm that LFU and its approximations (e.g., LRU-2) are resilient to mice-flow attacks but vulnerable to elephant-flow attacks; under medium-flow attacks, these policies still guarantee service for the largest few legitimate flows, thus achieving an intermediate performance. Moreover, we see that which policy performs the best will vary based on the attack strategy and the number of attack flows.

V. ATTACK-AWARE POLICY SELECTION

We further exploit the use of attack-resilient replacement policies as a second line of defense, in scenarios where efforts to prevent/detect attacks have failed and the cache cannot distinguish legitimate requests from malicious requests.

The results from Section IV-B suggest that *no single replacement policy can maximize the hit ratio for legitimate users in all the attack scenarios*. Therefore, the policy needs to be adapted based on the current level of attack, where the TTL approximations can provide valuable information.

Specifically, while the exact rates of attack flows $(\lambda_f)_{f \in F_a}$ are hard to estimate (because the cache does not know which flows are attack flows), it is often possible to estimate coarse parameters of the attack, such as the number of attack flows C_a

and their total rate Λ_a . For example, by comparing the current number and total rate of flows to the expected values from the history, we can use the surplus (if any) to estimate these parameters for a suspected DoS attack. From Section IV-A, we know that the optimal attack strategy under these estimated parameters is to send C_a flows of equal rate $\lambda_a := \Lambda_a/C_a$. Therefore, we can obtain a conservative estimate of the legitimate users' average hit ratio by identifying C_a of the current flows with rates around λ_a and a total rate around Λ_a as "attack flows" and considering the rest as legitimate flows.

Let F denote the current set of flows and F_a the estimated subset of attack flows. Let Π denote the set of candidate policies. We can use the TTL approximations to select the best policy in Π as follows:

- 1) for each candidate policy $\pi \in \Pi$, solve the characteristic equation $\sum_{f \in F} h^\pi(\lambda_f, T) = C$ for the characteristic time T under policy π ;
- 2) based on the calculated characteristic times, estimate the average hit ratio \bar{h}^π of the legitimate flows under each $\pi \in \Pi$ by (25), where $F_l := F \setminus F_a$;
- 3) select the policy π^* with the maximum \bar{h}^π .

Remark: The above method of estimating attack flows is not meant to accurately detect the attack flows; instead, we only use it to compute a conservative estimate of the hit ratio for legitimate flows, while the actual hit ratio can only be higher if the attack flows are different from our estimate (as long as there are no more than C_a attack flows of a total rate no more than Λ_a). Moreover, when the cache cannot maintain the exact flow rates $(\lambda_f)_{f \in F}$ (e.g., due to memory limitation), we can use approximations, e.g., computed by sketching [33].

VI. PERFORMANCE EVALUATION

We evaluate the proposed solutions on both synthetic and real request processes, in the scenario where the cache represents a flow table at an SDN switch. Functioning as a cache of flow rules from the controller, the flow table is particularly vulnerable to DoS attacks due to its small size as shown in [10], [21], [25], [22], [23], [24]. In this context, a "request" is an incoming packet, a "content" is a flow rule, and the access delay is the time for the switch to query the controller and install a new rule upon a table miss. "Hit ratios" in the sequel always refer to the hit ratios of legitimate flows.

A. Simulation Setting

We set the cache size $C = 1000$ according to the flow table size of commodity switches [34], and the average access delay $\bar{D} = 20$ ms according to the performance of such switches [35]. We set $q = 0.15$ for q -LRU. We generate attack flows as independent Poisson processes of rates to be specified later. We generate legitimate flows in two ways:

1) *Synthetic simulation:* To verify our theoretical predictions, we generate $|F_l| = 5000$ Poisson processes with total rate $\Lambda_l = 10$ packets/ms and a Zipf(α) popularity distribution with skewness $\alpha = 1$. Here, $|F_l|$ is set according to the maximum number of active flows (90% of the time) at a data center switch [36], Λ_l according to the average rate of

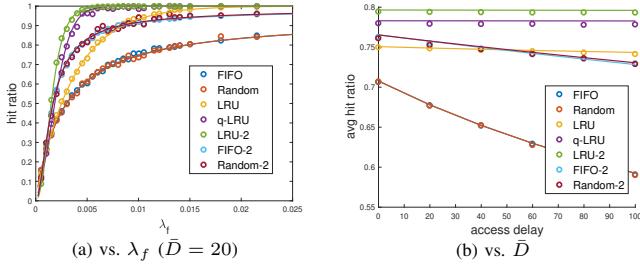


Fig. 7. Accuracy of TTL approximation (o: simulated; —: predicted).

the corresponding traces [37], and α according to the typical skewness of these traces.

2) *Trace-driven simulation*: To validate our findings made under the IRM assumption, we also use real traces as legitimate flows. To this end, we use the UNI2 dataset from [37], which contains 9 trace files, each containing 29,312–47,807 flows of a total rate between 9.84 and 11.31 packets/ms. From each file, we extract 10 traces of 10,000 packets from disjoint time periods with sufficiently many active flows. It is known [36] that these traces deviate from Poisson processes.

B. Results

1) *Accuracy of TTL Approximation*: To verify the accuracy of our TTL approximation formulas, we compare the simulated and the predicted hit ratios for each flow generated as in Section VI-A1 without any attack. The results in Fig. 7 (a) show that the prediction by our formulas is highly accurate under IRM. We further vary the access delay and evaluate the average hit ratio over all the flows in Fig. 7 (b). Besides verifying the accuracy of our formulas, this result also demonstrates the value of considering access delays, as ignoring such delays can cause significant overestimation of the hit ratios.

2) *Impact of DoS Attack*: Next, we evaluate the average hit ratio for legitimate users under DoS attacks. For Poisson traffic (Fig. 8), the TTL approximations accurately predict the performance for legitimate users under a wide range of attacks, thus validating our observations in Section IV-B (similar results hold under medium-flow attack, omitted for space). For the traces, as the flows and their rates vary from trace to trace, we plot the distribution of average hit ratios of all the traces under three representative attack strategies (Fig. 9 (a–c)). We see that while the prediction is no longer exact, it captures important trends: (i) simple indiscriminate policies (e.g., FIFO, Random) are resilient to elephant-flow attacks but vulnerable to other attacks; (ii) highly discriminative policies (e.g., LRU-2) are resilient to mice/medium-flow attacks but vulnerable to elephant-flow attacks; (iii) two-staged policies with indiscriminate eviction rules (e.g., FIFO-2, Random-2) are resilient to both mice-flow and elephant-flow attacks but vulnerable to medium-flow attacks with suitable rates. These results validate our observation in Section V that no single policy can optimize the performance in all the attack scenarios.

3) *Performance of Policy Selection*: Since Fig. 9 (a–c) already show that the TTL approximations can guide us to the best policy under static attacks, we now focus on time-varying attacks. To this end, we simulate a hybrid attack,

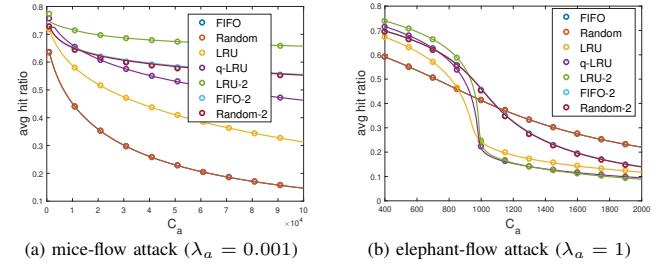


Fig. 8. Impact of DoS attack on synthetic traffic (o: simulated; —: predicted).

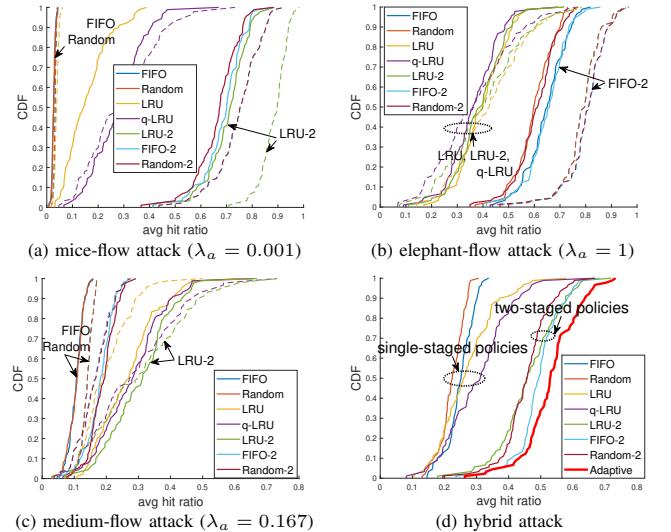


Fig. 9. Impact of DoS attack on traces under total attack rate 1000 (solid: simulated; dashed: predicted).

where for each trace, attack traffic is sent according to the mice-flow attack in Fig. 9 (a) for the first 1/3 of the trace, the medium-flow attack in Fig. 9 (c) for the second 1/3 of the trace, and the elephant-flow attack in Fig. 9 (b) for the last 1/3 of the trace; similar results hold under other orders of applying these attack strategies (omitted for space). The intuition is to take advantage of the fact that none of the policies is resilient against all these attacks. Fig. 9 (d) shows the distribution of the average hit ratio over all the traces. We see that (i) the adaptive policy selection scheme achieves a better performance than any single policy under the hybrid attack by combining the strengths of different policies, and (ii) two-staged policies, especially FIFO-2, are more robust than single-staged policies.

VII. CONCLUSION

Inspired by empirical studies that showed poor performance of normally good replacement policies under DoS attacks, we performed a systematic study of the attack resilience of a set of state-of-the-art policies using the tool of TTL approximations. After incorporating access delays into these approximations, we used them to design the optimal attack strategy against each policy and develop an attack-aware policy selection scheme. Our case study in SDN validated our solutions, particularly that the proposed policy selection scheme can effectively improve the attack resilience of the cache. Our results also identified two-staged policies, especially FIFO-2, as an attack-oblivious option with relatively good resilience.

REFERENCES

- [1] S. Podlipnig and L. Böszörmenyi, "A survey of web cache replacement strategies," *ACM Comput. Surv.*, vol. 35, no. 4, p. 374–398, Dec. 2003. [Online]. Available: <https://doi.org/10.1145/954339.954341>
- [2] W. Ali, S. M. Shamsuddin, A. S. Ismail *et al.*, "A survey of web caching and prefetching," *Int. J. Advance. Soft Comput. Appl.*, vol. 3, no. 1, pp. 18–44, 2011.
- [3] Kin-Young Wong, "Web cache replacement policies: a pragmatic approach," *IEEE Network*, vol. 20, no. 1, pp. 28–34, 2006.
- [4] S. Basu, A. Sundarajan, J. Ghaderi, S. Shakottai, and R. Sitaraman, "Adaptive ttl-based caching for content delivery," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1063–1077, June 2018.
- [5] I. Abdullahi, S. Arif, and S. Hassan, "Survey on caching approaches in information centric networking," *Journal of Network and Computer Applications*, vol. 56, pp. 48–59, 2015.
- [6] S. Alouf, N. Choungmo Fofack, and N. Nedkov, "Performance models for hierarchy of caches: Application to modern DNS caches," *Performance Evaluation*, vol. 97, pp. 57–82, Mar. 2016, performance Evaluation Methodologies and Tools: Selected Papers from VALUETOOLS 2013. [Online]. Available: <https://hal.inria.fr/hal-01258189>
- [7] N. Gast and B. Van Houdt, "Asymptotically exact TTL-approximations of the cache replacement algorithms LRU(m) and h-LRU," in *2016 28th International Teletraffic Congress (ITC 28)*, vol. 01, 2016, pp. 157–165.
- [8] M. Garetto, E. Leonardi, and V. Martina, "A unified approach to the performance analysis of caching systems," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 1, no. 3, May 2016.
- [9] Y. Gao, L. Deng, A. Kuzmanovic, and Y. Chen, "Internet cache pollution attacks and countermeasures," in *Proceedings of the 2006 IEEE International Conference on Network Protocols*, 2006, pp. 54–64.
- [10] M. Yu, T. He, P. McDaniel, and Q. K. Burke, "Flow table security in SDN: Adversarial reconnaissance and intelligent attacks," in *INFOCOM*, 2020.
- [11] D. S. Berger, P. Gland, S. Singla, and F. Ciucu, "Exact analysis of ttl cache networks," *Performance Evaluation*, vol. 79, pp. 2–23, 2014.
- [12] "Open vSwitch 2.14.90 Documentation," <https://docs.openvswitch.org/en/latest/>.
- [13] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *IEEE journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1305–1314, 2002.
- [14] G. Bianchi, A. Detti, A. Caponi, and N. Blefari Melazzi, "Check before storing: What is the performance price of content integrity verification in lru caching?" *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, p. 59–67, Jul. 2013. [Online]. Available: <https://doi.org/10.1145/2500098.2500106>
- [15] B. Jiang, P. Nain, and D. Towsley, "On the convergence of the TTL approximation for an LRU cache under independent stationary request processes," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 3, no. 4, September 2018.
- [16] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance," in *Proceedings of the 24th International Teletraffic Congress*, 2012.
- [17] A. Dabirmoghaddam, M. Dehghan, and J. J. Garcia-Luna-Aceves, "Characterizing interest aggregation in content-centric networks," in *IFIP Networking*, May 2016.
- [18] M. Dehghan, B. Jiang, A. Dabirmoghaddam, and D. Towsley, "On the analysis of caches with pending interest tables," in *ICN*, September 2015.
- [19] E. G. AbdAllah, H. S. Hassanein, and M. Zukernine, "A survey of security attacks in information-centric networking," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1441–1454, 2015.
- [20] Y. Zhou, K. Chen, J. Zhang, J. Leng, and Y. Tang, "Exploiting the vulnerability of flow table overflow in software-defined networks: Attack model, evaluation, and defense," *Security and Communication Networks*, pp. 1–15, January 2018.
- [21] J. Cao, M. Xu, Q. Li, K. Sun, Y. Yang, and J. Zheng, "Disrupting sdn via the data plane: A low-rate flow table overflow attack," in *SECURECOMM*, 2017.
- [22] Y. Qian, W. You, and K. Qian, "Openflow flow table overflow attacks and countermeasures," in *IEEE EuCNC*, 2016.
- [23] B. Yuan, D. Zou, S. Yu, H. Jin, W. Qiang, and J. Shen, "Defending against flow table overloading attack in software-defined networks," *IEEE Transactions on Services Computing*, vol. 12, no. 2, pp. 231–246, March-April 2019.
- [24] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks," in *ACM CCS*, November 2013.
- [25] J. Weekes and S. Nagaraja, "Controlling your neighbour's bandwidth for fun and for profit," in *Security Protocols*, 2017.
- [26] H. M. Sun, W. H. Chang, S. Y. Chang, and Y. H. Lin, "DepenDNS: Dependable mechanism against DNS cache poisoning," in *Cryptology and Network Security*. New York, NY, USA: Springer-Verlag, 2009, p. 174–188.
- [27] C. Ghali, G. Tsudik, and E. Uzun, "Needle in a haystack: Mitigating content poisoning in named-data networking," in *SENT Workshop at NDSS*, 2014.
- [28] T. Kamimoto, K. Mori, S. Umeda, Y. Ohata, and H. Shigeno, "Cache protection method based on prefix hierarchy for content-oriented network," in *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2016, pp. 417–422.
- [29] Z. Tan, A. Jamdagni, X. He, P. Nanda, and R. P. Liu, "A system for denial-of-service attack detection based on multivariate correlation analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 447–456, 2014.
- [30] H. Park, I. Widjaja, and H. Lee, "Detection of cache pollution attacks using randomness checks," in *2012 IEEE International Conference on Communications (ICC)*, 2012, pp. 1096–1100.
- [31] M. Xie, I. Widjaja, and H. Wang, "Enhancing cache robustness for content-centric networking," in *IEEE INFOCOM*, 2012.
- [32] N. Choungmo-Fofack, M. Dehghan, D. Towsley, M. Badov, and D. L. Goeckel, "On the performance of general cache networks," in *ValueTools*, December 2014, p. 106–113.
- [33] Y. Fu, D. Li, S. Shen, Y. Zhang, and K. Chen, "Clustering-preserving network flow sketching," in *INFOCOM*, 2020.
- [34] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, January 2015.
- [35] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "Cacheflow: Dependency-aware rule-caching for software-defined networks," in *SOSR*, 2016.
- [36] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010.
- [37] T. Benson, "Data set for IMC 2010 data center measurement," http://pages.cs.wisc.edu/~tbenson/IMC10_Data.html.