Attack Resilience of Cache Replacement Policies: A Study Based on TTL Approximation

Tian Xie, *Student Member, IEEE*, Namitha Nambiar, Ting He, *Senior Member, IEEE*, and Patrick McDaniel, *Fellow, IEEE, ACM*

***Abstract*— Caches are pervasively used in communication networks to speed up content 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 cache pollution 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. Motivated by these results, we implement selected policies as well as policy adaptation in an open-source SDN switch to manage flow rule replacement, which is shown to notably improve its resilience to pollution attacks.**

***Index Terms*— Cache replacement, access delay, cache pollution 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) [2], Content Delivery Networks (CDNs) [3], Information Centric Networking (ICN) [4], and Domain Name System (DNS) [5]. 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 multiple stages [6], [7]. However, most existing works only considered the performance in benign environments, where all the requests are from legitimate users.

Meanwhile, empirical studies in [8], [9] revealed that a policy with superior performance in benign environments can perform poorly under a type of DoS attacks, referred to as *cache pollution attack*, that flood the cache with requests for unpopular contents, thus denying the legitimate users their chance to receive service from the cache [10]. For example, the Least Frequently Used (LFU) policy that is known to be optimal in the benign environment under the Independent Reference Model (IRM) [7] performs worse than LRU under such attacks [8], [9], which is in turn worse than FIFO [9].

Motivated by these observations, we perform a comprehensive study of the *attack resilience of cache replacement policies* in the context of communication networks. As an important aspect of general network resilience [11], attack resilience is the ability to maintain an acceptable level of service in the presence of attacks, where our focus is on the ability of cache replacement policies to maintain the hit ratio for legitimate requests in the presence of pollution attacks. Performance under attacks is important because (i) an attack can be long-lasting, and (ii) there can be frequent intermittent attacks, both causing the network to operate under attacks for a significant fraction of time. To quantify the attack resilience of cache replacement policies, we analyze the hit ratios of representative policies using the tool of Time-to-Live (TTL) approximations while incorporating content access delays. Such approximations not only allow us to explain how pollution attacks affect the hit ratio of legitimate requests, but also shed light on the attack strategies and the defenses.

Manuscript received 11 June 2021; revised 17 October 2021 and 5 March 2022; accepted 16 April 2022; approved by IEEE/ACM Transactions on Networking Editor Y. Liu. Date of publication 27 May 2022; date of current version 20 December 2022. This work was supported in part by the National Science Foundation under Award CNS-1946022, in part by the U.S. Army Combat Capabilities Development Command Army Research Laboratory, and in part by the U.S. Army Research Laboratory (ARL) Cyber Security Collaborative Research Alliance (CRA) under Cooperative Agreement W911NF-13-2-0045. A preliminary version of this work was presented at INFOCOM'21 [DOI: 10.1109/INFOCOM42981.2021.9488697]. *(Corresponding author: Tian Xie.)*

Tian Xie, Ting He, and Patrick McDaniel are with the Department of Computer Science and Engineering, The Pennsylvania State University, University Park, State College, PA 16802 USA (e-mail: tbx5027@psu.edu; tzh58@psu.edu; pdm12@psu.edu).

Namitha Nambiar is with LinkedIn, Sunnyvale, CA 94085 USA (e-mail: namitha.snambiar95@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2022.3171720, provided by the authors.

Digital Object Identifier 10.1109/TNET.2022.3171720

A. Related Work

1) 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 [12]. 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) [2]. However, when maintaining consistency with the origin server is important, e.g., in DNS and WWW, TTL-based policies are popular [12]. In cases such as SDN, a combination of both types of policies is used [13]. 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 [6], various approximations have been developed, most notably the TTL approximation that models capacity-driven policies by TTL-based policies [14]. This idea has been used to predict the hit ratio for a number of capacity-driven policies, including FIFO, Random, LRU, and their variations [6], [7], [15]. The request processes under which these approximations apply have also been generalized from Poisson processes (i.e., IRM) [14] to renewal processes [7], Markov processes [6], and general stationary processes [16]. Besides known to be numerically accurate, TTL approximations are also shown to be asymptotically exact for large caches [6], [16], [17].

Application of cache replacement policies have also been studied in various systems, e.g., WWW [2], CDN [3], ICN [4], and DNS [5]. Many of these systems employ a network of interconnected caches, for which analytical results have been obtained under TTL approximations [5], [12], [18].

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 [19], 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.

2) Attacks on Caches and Defenses: Caches have been the common targets of malicious attacks. In the networking context, caches can be used to extract private information [9], [20], [21], but the focus has been on degrading the cache performance by overwhelming its capacity [20], [22]–[25] or occupying it with unpopular contents [8], [9], [20], [26], 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., [27] for DNS, [28], [29] for ICN, [21]–[25] for SDN) or detecting attacks [8], [30], [31]. In contrast, we aim at understanding the *attack resilience of the cache itself*. Although attack resilience of caches has been briefly discussed in [26], [32], [32] only considered one attack strategy (similar to mice-flow attack considered in Section IV-A), and [26] 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 five-fold:

1) We extend the TTL approximation to incorporate the delays for the cache to obtain missing contents for a set of policies known to have superior performance in benign environments [7]. We further discuss the impact of idle/hard timeouts on TTL approximation and extend our analysis to capture such impact. These results advance the state of the art on TTL approximation, 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 under different attack strategies can be different, we propose a scheme to adapt the replacement policy based on coarse parameters of the attack.

4) Treating the flow table in an SDN switch as the cache of interest, we perform a simulation-based performance evaluation. 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.

5) We implement selected policies as well as runtime policy adaptation to manage flow rule replacement in Open vSwitch. Our experiments in a virtual SDN based on real traces show that the added policies have notably better resilience than the original rule replacement policy under certain attacks, and the proposed policy adaptation scheme can further improve the resilience to time-varying attacks.

Roadmap. We will formulate our problem in Section II, present our TTL approximation results in Section III, analyze the optimal attack strategy and its impact in Section IV, present our attack-aware policy selection scheme in Section V, present our simulation results in Section VI and our experiment results in Section VII, and conclude the paper in Section VIII. **All appendices can be found in the supplementary file.**

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 pollution 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 $\Pr\{Y_i \le y\} = G_f(y)$ for all $y \ge 0$. Let $\tilde{G}_f(y|t)$ denote the distribution function of the *excess life* at time t, i.e., if Γ_t is the time from t to the next arrival, then $\Pr\{\Gamma_t \le y\} = \tilde{G}_f(y|t)$ for all $y \ge 0$. 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) [7], and some of the above policies can approximate LFU without requiring prior knowledge. Specifically, q-LRU tends to LFU as $q \to 0$, and LRU-k tends to LFU as $k \to \infty$, with much of the performance gain achieved at $k = 2$ [7].

While traditional cache performance analysis assumes that a content is immediately available at the cache after a miss,[1] 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 pollution attacks, and develop new policies with better attack resilience. Our secondary objective is to advance the state of the art on TTL approximation by incorporating access delays and timeouts.

III. TTL Approximation With Access Delay

Traditional TTL approximation formulas [7] 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 [19] 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 [19]:

• FIFO: A FIFO cache can be modeled as a TTL cache with constant non-reset timers of timeout T [33]. Under Poisson arrivals, the hit ratio and the occupancy probability for content f are

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

Under renewal arrivals, the hit ratio is

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

and the occupancy probability is

$$o_f^{FIFO} = \frac{T}{\bar{D} + T + \mathbb{E}[\Gamma_f(D+T)]}, \tag{3}$$

where

$$\mathbb{E}[\Gamma_f(D+T)] = \mathbb{E}\left[\int_0^\infty \left(1 - \tilde{G}_f(t|D+T)\right) dt\right] \tag{4}$$

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

is the expected excess life for the arrival process of requests for content f at time $D+T$. The expectations in (2) and (4) are over D.

• Random: A Random cache can be modeled as a TTL cache with exponentially distributed non-reset timers with mean timeout $\bar{T}$ [33]. Under Poisson arrivals, the hit ratio and the occupancy probability for content f are

$$h_f^{\text{Random}} = o_f^{\text{Random}} = \frac{\lambda_f \bar{T}}{1+\lambda_f(\bar{D}+\bar{T})}, \tag{5}$$

which is identical to (1) except that T is replaced by $\bar{T}$. Under renewal arrivals, the hit ratio and the occupancy probability are the same as (2) and (3), respectively, except that the expectations are 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 [33]. Under Poisson arrivals, the hit ratio and the occupancy probability for content f are

$$h_f^{LRU} = o_f^{LRU} = \frac{e^{\lambda_f T}-1}{\lambda_f \bar{D} + e^{\lambda_f T}}. \tag{6}$$

Under renewal arrivals, this hit ratio is

$$h_f^{LRU} = \frac{\mathbb{E}[\tilde{G}_f(T|D)]}{(1-G_f(T))(1+\mathbb{E}[m_f(D)])+\mathbb{E}[\tilde{G}_f(T|D)]}, \tag{7}$$

and the occupancy probability is

$$o_f^{LRU} = \frac{\lambda_f \mathbb{E}[\Gamma_f(D)] + \mathbb{E}[N_f] - \lambda_f \mathbb{E}[\Gamma_f(T_e)]}{1+\mathbb{E}[m_f(D)]+\mathbb{E}[N_f]}, \tag{8}$$

where $\mathbb{E}[\Gamma_f(D)]$ and $\mathbb{E}[\Gamma_f(T_e)]$ are defined similarly as (4) (T_e: the content eviction time from the beginning of a renewal period), and

$$\mathbb{E}[N_f] = \frac{\mathbb{E}[\tilde{G}_f(T|D)]}{1-G_f(T)} \tag{9}$$

is the expected number of hits per renewal period. The expectations are over D and T_e.

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

$$\sum_{f\in F} o_f = C. \tag{10}$$

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

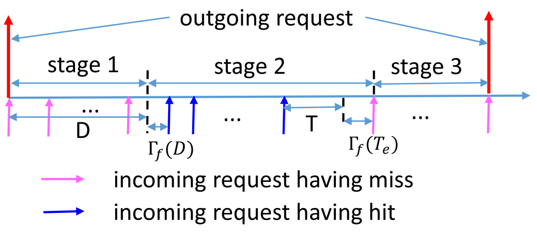


Fig. 1. Renewal period under q-LRU.

$$h_f = \frac{\mathbb{E}[X_2]}{\mathbb{E}[X_1]+\mathbb{E}[X_2]+\mathbb{E}[X_3]}. \tag{11}$$

Furthermore, let $Y_{i,j}$ denote the inter-arrival time between the j-th request in stage i and the next request. By definition of the renewal period, its duration equals $D+\Gamma_f(D)+\sum_{i=2}^{3}\sum_{j=1}^{X_i} Y_{i,j}$, during which content f is cached for time $\Gamma_f(D)+\sum_{j=1}^{X_2} Y_{2,j} - \Gamma_f(T_e)$, where $\Gamma_f(D)$ denotes the time from the arrival of the requested content to the first hit (i.e., the excess life at time D, if the period starts from time 0), and $\Gamma_f(T_e)$ denotes the time from the eviction of the content to the next request (i.e., the excess life at the eviction time T_e). The occupancy probability is thus

$$\begin{aligned} o_f &= \frac{\mathbb{E}[\Gamma_f(D)+\sum_{j=1}^{X_2} Y_{2,j} - \Gamma_f(T_e)]}{\mathbb{E}[D+\Gamma_f(D)+\sum_{i=2}^{3}\sum_{j=1}^{X_i} Y_{i,j}]} \\ &= \frac{\mathbb{E}[\Gamma_f(D)] + \frac{1}{\lambda_f}\mathbb{E}[X_2] - \mathbb{E}[\Gamma_f(T_e)]}{\bar{D}+\mathbb{E}[\Gamma_f(D)]+\frac{1}{\lambda_f}\mathbb{E}[X_2]+\frac{1}{\lambda_f}\mathbb{E}[X_3]}, \end{aligned} \tag{12}$$

where we have applied Wald's identity to $\sum_{j=1}^{X_2} Y_{2,j}$ and $\sum_{j=1}^{X_3} Y_{3,j}$ as X_2 is a stopping time for $\{Y_{2,1}, Y_{2,2}, \ldots\}$ ($X_2 \le n$ if and only if $\exists 1 \le j \le n$ such that $Y_{2,j} > T$) and X_3 is independent of $\{Y_{3,1}, Y_{3,2}, \ldots\}$.

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 [33], 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,

$$\Pr\{X_2 = n\} = (1-e^{-\lambda_f T})^n e^{-\lambda_f T}, \quad n = 0,1,\ldots, \tag{13}$$

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

$$\Pr\{X_3 = m\} = (1-q)^m q, \quad m = 0, 1, \ldots, \tag{14}$$

and $\mathbb{E}[X_3] = (1-q)/q$. Plugging these results into (11) yields

$$h_f^{q-LRU} = \frac{e^{\lambda_f T} - 1}{\lambda_f \bar{D} + e^{\lambda_f T} + \frac{1-q}{q}}, \tag{15}$$

which reduces to (6) as $q \to 1$ as expected. The parameter T in (15) can be solved from $\sum_{f\in F} h_f^{q-LRU} = C$ as $o_f^{q-LRU} = h_f^{q-LRU}$ under Poisson arrivals.

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,

$$\Pr\{X_2 = n|D\} = \begin{cases} 1 - \tilde{G}_f(T|D) & if n = 0, \\ \tilde{G}_f(T|D) G_f(T)^{n-1}(1 - G_f(T)) & o.w., \end{cases} \tag{16}$$

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. Plugging these results into (11) yields

$$h_f^{q-LRU} = \frac{\frac{\mathbb{E}[\tilde{G}_f(T|D)]}{1-G_f(T)}}{1 + \mathbb{E}[m_f(D)] + \frac{\mathbb{E}[\tilde{G}_f(T|D)]}{1-G_f(T)} + \frac{1-q}{q}}, \tag{17}$$

where the only unknown parameter is T. Based on (12), T can be obtained by solving (10) for

$$o_f^{q-LRU} = \frac{\mathbb{E}[\Gamma_f(D)] + \frac{\mathbb{E}[\tilde{G}_f(T|D)]}{\lambda_f(1-G_f(T))} - \mathbb{E}[\Gamma_f(T_e)]}{\bar{D} + \mathbb{E}[\Gamma_f(D)] + \frac{\mathbb{E}[\tilde{G}_f(T|D)]}{\lambda_f(1-G_f(T))} + \frac{1-q}{\lambda_f q}}. \tag{18}$$

Discussion: Predicting the hit ratio using the TTL approximation for general renewal processes is mainly complicated by the need of computing the expected excess life. In particular, $\mathbb{E}[\Gamma_f(T_e)]$ in general depends on the distribution of the eviction time T_e, which is not analytically tractable as $T_e = D + \Gamma_f(D) + \sum_{j=1}^{X_2-1} Y_{2,j} + T$. Therefore, as in the literature, closed-form formulas have only been obtained for Poisson processes.

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 [7]. Below, we give detailed analysis for $k = 2$, and defer the general case to Appendix.A.

As illustrated in Fig. 2, each renewal period of cache 2 (the real cache) is the time between consecutive requests

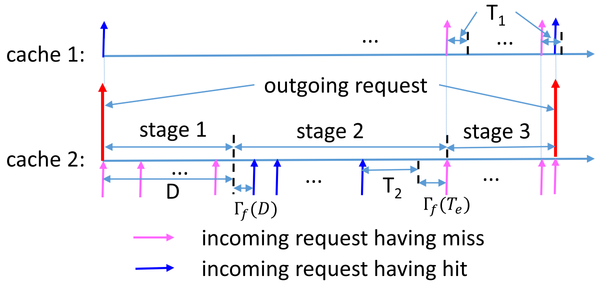


Fig. 2. Renewal period under LRU-2.

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 [33] 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 [7], [34], 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,

$$\Pr\{X_3 - 1 = m\} = e^{-m\lambda_f T_1}(1 - e^{-\lambda_f T_1}), \; m \geq 0, \tag{19}$$

and hence $\mathbb{E}[X_3] = 1/(1 - e^{-\lambda_f T_1})$. Plugging these results into (11) yields

$$h_f^{LRU-2} = \frac{e^{\lambda_f T_2} - 1}{\lambda_f \bar{D} + e^{\lambda_f T_2} + \frac{1}{1-e^{-\lambda_f T_1}}}. \tag{20}$$

Here, T_1 is the solution to the characteristic equation of cache 1: $\sum_{f\in F} h_f^{LRU} = C_1$ (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: $\sum_{f\in F} h_f^{LRU-2} = C$, where we have used the PASTA property of Poisson processes.

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,

$$\Pr\{X_3 - 1 = m\} = (1 - G_f(T_1))^m G_f(T_1), \; m \geq 0, \tag{21}$$

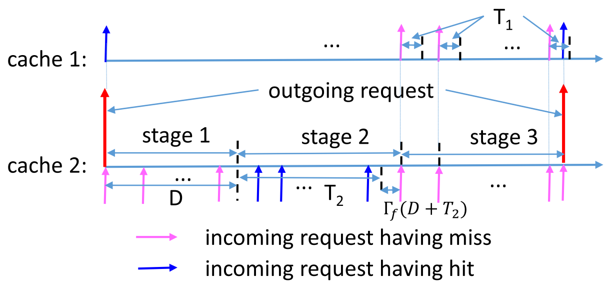


Fig. 3. Renewal period under FIFO-2.

and hence $\mathbb{E}[X_3] = 1/G_f(T_1)$. Plugging these results into (11) yields

$$h_f^{LRU-2} = \frac{\frac{\mathbb{E}[\tilde{G}_f(T_2|D)]}{1-G_f(T_2)}}{1+\mathbb{E}[m_f(D)] + \frac{\mathbb{E}[\tilde{G}_f(T_2|D)]}{1-G_f(T_2)} + \frac{1}{G_f(T_1)}}, \tag{22}$$

where T_1 and T_2 are the characteristic times of cache 1 and cache 2, respectively.

Furthermore, by similar arguments as in Section III-B, the occupancy probability in cache 2 equals

$$o_f^{LRU-2} = \frac{\mathbb{E}[\Gamma_f(D)] + \frac{1}{\lambda_f}\mathbb{E}[X_2] - \mathbb{E}[\Gamma_f(T_e)]}{\bar{D} + \mathbb{E}[\Gamma_f(D)] + \frac{1}{\lambda_f}\mathbb{E}[X_2] + \frac{1}{\lambda_f}\mathbb{E}[X_3]} \tag{23}$$

$$= \frac{\mathbb{E}[\Gamma_f(D)] + \frac{\mathbb{E}[\tilde{G}_f(T_2|D)]}{\lambda_f(1-G_f(T_2))} - \mathbb{E}[\Gamma_f(T_e)]}{\bar{D} + \mathbb{E}[\Gamma_f(D)] + \frac{\mathbb{E}[\tilde{G}_f(T_2|D)]}{\lambda_f(1-G_f(T_2))} + \frac{1}{\lambda_f G_f(T_1)}}, \tag{24}$$

where we can apply Wald's identity because X_2 is a stopping time for $\{Y_{2,1}, Y_{2,2}, \ldots\}$ as in Section III-B and X_3 is a stopping time for $\{Y_{3,1}, Y_{3,2}, \ldots\}$ ($X_3 \leq n$ if and only if $\exists 1 \leq j \leq n$ such that $Y_{3,j} \leq T_1$). We can solve T_1 by plugging the occupancy probability in cache 1, given by (8) for $D = 0$, into $\sum_{f\in F} o_f^{LRU} = C_1$. We can then solve T_2 by plugging (24) (which now only has T_2 as an unknown parameter) into (10).

D. TTL Approximation for FIFO-2

Our analysis in Section III-C extends naturally to other two-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 [33], 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) [33]. 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) [7], 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,

$$\Pr\{X_3 = m\} = \begin{cases} \frac{\lambda_f T_1}{1+\lambda_f T_1} & if m = 0, \\ \frac{e^{-(m-1)\lambda_f T_1}(1-e^{-\lambda_f T_1})}{1+\lambda_f T_1} & if m > 0, \end{cases} \tag{25}$$

and hence $\mathbb{E}[X_3] = 1/[(1+\lambda_f T_1)(1-e^{-\lambda_f T_1})]$. Plugging these results into (11) yields

$$h_f^{FIFO-2} = \frac{\lambda_f T_2}{1+\lambda_f(\bar{D}+T_2) + \frac{1}{(1+\lambda_f T_1)(1-e^{-\lambda_f T_1})}}. \tag{26}$$

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

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

$$\Pr\{X_3 = m\} = \begin{cases} \dfrac{m_f(T_1)}{1+m_f(T_1)} & if m = 0, \\ \dfrac{1}{1+m_f(T_1)}(1-G_f(T_1))^{m-1}G_f(T_1) & o.w., \end{cases} \tag{27}$$

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 (11) yields

$$h_f^{FIFO-2} = \frac{\mathbb{E}[m_f(D+T_2)] - \mathbb{E}[m_f(D)]}{1+\mathbb{E}[m_f(D+T_2)] + \frac{1}{(1+m_f(T_1))G_f(T_1)}}, \tag{28}$$

where T_1 and T_2 are characteristic times of cache 1 and cache 2, respectively.

To computer the characteristic times, we note that the expected duration of a renewal period equals

$$\mathbb{E}\left[D + T_2 + \Gamma_f(D+T_2) + \sum_{j=1}^{X_3} Y_{3,j}\right]$$
$$= \bar{D} + T_2 + \mathbb{E}[\Gamma_f(D+T_2)] + \frac{1}{\lambda_f}\mathbb{E}[X_3], \tag{29}$$

where we have applied Wald's identity. During each period, the content occupies cache 2 for time T_2. Thus, the occupancy probability in cache 2 is given by

$$o_f^{FIFO-2} = \frac{T_2}{\bar{D}+T_2+\mathbb{E}[\Gamma_f(D+T_2)]+\frac{1}{\lambda_f}\mathbb{E}[X_3]} \quad (30)$$

$$= \frac{T_2}{\bar{D}+T_2+\mathbb{E}[\Gamma_f(D+T_2)]+\frac{1}{\lambda_f(1+m_f(T_1))G_f(T_1)}}. \quad (31)$$

We can first solve T_1 from $\sum_{f\in F} o_f^{FIFO} = C_1$, where $o_f^{FIFO} = T_1/\left(T_1+\mathbb{E}[\Gamma_f(T_1)]\right)$ is the occupancy probability in cache 1, and then solve T_2 by plugging (31) into (10).

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 [33]. 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 variables,[2] 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 [7], 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 (5). 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,

$$\Pr\{X_3 = m\} = \left(\frac{1}{1+\lambda_f\bar{T}_1}\right)^m \left(\frac{\lambda_f\bar{T}_1}{1+\lambda_f\bar{T}_1}\right),\ m \geq 0, \quad (32)$$

which implies that $\mathbb{E}[X_3] = 1/(\lambda_f\bar{T}_1)$. Plugging these results into (11) yields

$$h_f^{\text{Random-2}} = \frac{\lambda_f\bar{T}_2}{1+\lambda_f(\bar{D}+\bar{T}_2)+\frac{1}{\lambda_f\bar{T}_1}}, \quad (33)$$

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 $\sum_{f\in F} h_f^{\text{Random-2}} = C$. In the special case of exponentially distributed access delays, a more accurate TTL approximation can be computed without the independence assumption; see details in Appendix.A.

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

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 $m \geq 1$,

$$\Pr\{X_3 = m\} = \frac{\mathbb{E}[G_f(T_1)]}{1+\mathbb{E}[m_f(T_1)]}\left(1-\mathbb{E}[G_f(T_1)]\right)^{m-1}, \quad (34)$$

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 (11) yields

$$h_f^{\text{Random-2}} = \frac{\mathbb{E}[m_f(D+T_2)] - \mathbb{E}[m_f(D)]}{1+\mathbb{E}[m_f(D+T_2)]+\frac{1}{(1+\mathbb{E}[m_f(T_1)])\mathbb{E}[G_f(T_1)]}}, \quad (35)$$

where the only unknown parameters are the means $\bar{T}_i$ of T_i for $i = 1$ and 2.

Moreover, by similar arguments as in Section III-D2, the occupancy probability in cache 2 equals

$$o_f^{\text{Random}-2} \quad (36)$$
$$= \frac{\bar{T}_2}{\bar{D}+\bar{T}_2+\mathbb{E}[\Gamma_f(D+T_2)]+\frac{1}{\lambda_f}\mathbb{E}[X_3]}$$
$$= \frac{\bar{T}_2}{\bar{D}+\bar{T}_2+\mathbb{E}[\Gamma_f(D+T_2)]+\frac{1}{\lambda_f(1+\mathbb{E}[m_f(T_1)])\mathbb{E}[G_f(T_1)]}}. \quad (37)$$

We can solve $\bar{T}_1$ from $\sum_{f\in F} o_f^{\text{Random}} = C_1$, where $o_f^{\text{Random}} = \bar{T}_1/\left(\bar{T}_1+\mathbb{E}[\Gamma_f(T_1)]\right)$ is the occupancy probability in cache 1. We can then solve $\bar{T}_2$ by plugging (37) into (10).

Remark: Although seemingly similar, (28) and (35) (or (31) and (37)) differ subtly in that T_i ($i = 1, 2$) is treated as a constant for FIFO-2 but a random variable for Random-2, 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 (26) and (33). This result is in contrast to the previous result that FIFO and Random have the same hit ratio under IRM [7].

F. TTL Approximation with Explicit Timeouts

In some application scenarios (e.g., SDN), cached contents are subject to explicit hard/idle timeouts, in addition to evictions caused by newly inserted contents. Hereby we discuss the impact of such explicit timeouts on the TTL approximation. Denote the idle timeout by τ_I and the hard timeout by τ_H. A cached content will be evicted if (i) it has been in the cache for τ_H time, (ii) it has not been requested for τ_I time, or (iii) it is selected to make room for a new content by the replacement policy, whichever occurs first. We will focus on the case of Poisson arrivals, which is already challenging to analyze as explained below.

Recall that the hit ratio for a given content f can be computed by (11) based on the expected number of arrivals

$\mathbb{E}[X_i]$ in stage $i = 1, 2, 3$ in each renewal period. By definition, $\mathbb{E}[X_1]$ only depends on the access delay and the arrival process, and is thus invariant to the timeouts. For a Poisson request process of rate λ_f, $\mathbb{E}[X_1] = 1 + \lambda_f \bar{D}$ as analyzed before (including the arrival at the beginning of the renewal period). Meanwhile, $\mathbb{E}[X_3]$ only depends on when the replacement policy decides to cache the missing content. For FIFO, Random, and LRU, $\mathbb{E}[X_3] = 0$ (recall that X_3 is the number of misses *before* the cache requests the missing content from the origin server). For q-LRU, $\mathbb{E}[X_3]$ only depends on the parameter q of the policy, and thus remains $\mathbb{E}[X_3] = (1-q)/q$. For the two-staged policies (LRU-2, FIFO-2, and Random-2), $\mathbb{E}[X_3]$ only depends on when there is a hit in the virtual cache. As the virtual cache is part of the replacement policy and not subject to externally-imposed timeouts, our previous analysis of $\mathbb{E}[X_3]$ remains valid. Thus, the only value in (11) affected by explicit timeouts is $\mathbb{E}[X_2]$.

Under the TTL approximation, the explicit timeouts τ_I and τ_H turn the (real) cache into a *hybrid TTL cache* with both a reset timer $\tilde{\tau}_I$ and a non-reset timer $\tilde{\tau}_H$. Under FIFO/Random eviction (i.e., FIFO, Random, FIFO-2, Random-2), $\tilde{\tau}_H = \min(T, \tau_H)$ and $\tilde{\tau}_I = \tau_I$. Under LRU eviction (i.e., LRU, q-LRU, LRU-2), $\tilde{\tau}_H = \tau_H$ and $\tilde{\tau}_I = \min(T, \tau_I)$. Let t_i denote the i-th interarrival time since the beginning of stage 2. As X_2 denotes the number of hits, i.e., the number of consecutive arrivals before either timer expires, we have

$$\Pr\{X_2 = n\} = \Pr\Big\{(t_1, \ldots, t_n \le \tilde{\tau}_I) \wedge (\sum_{i=1}^{n} t_i \le \tilde{\tau}_H) \wedge (t_{n+1} > \tilde{\tau}_I \vee \sum_{i=1}^{n+1} t_i > \tilde{\tau}_H)\Big\}, \quad (38)$$

where "$\wedge$" denotes AND and "$\vee$" denotes OR. By the Bayesian rule, (38) can be decomposed into

$$\Pr\{X_2 = n\} = \Pr\{A\} \cdot \Pr\{B|A\} \cdot \Big(\Pr\{t_{n+1} > \tilde{\tau}_I | A, B\} + \Pr\{\sum_{i=1}^{n+1} t_i > \tilde{\tau}_H, t_{n+1} \le \tilde{\tau}_I | A, B\}\Big), \quad (39)$$

where $A := t_1, \ldots, t_n \le \tilde{\tau}_I$, $B := \sum_{i=1}^{n} t_i \le \tilde{\tau}_H$. For a Poisson request process of rate λ_f, $\Pr\{A\} = (1 - e^{-\lambda_f \tilde{\tau}_I})^n$, and $\Pr\{t_{n+1} > \tilde{\tau}_I | A, B\} = e^{-\lambda_f \tilde{\tau}_I}$ (since t_{n+1} is independent of $t_1, \ldots, t_n$). Define $M_f(\tau)$ as the counting process of a renewal process with truncated exponential interarrival times with PDF $\lambda_f e^{-\lambda_f t}/(1 - e^{-\lambda_f \tilde{\tau}_I})$ for $0 \le t \le \tilde{\tau}_I$ and zero elsewhere. Then $\Pr\{B|A\} = \Pr\{M_f(\tilde{\tau}_H) \ge n\}$, and

$$\begin{aligned}\Pr\{\sum_{i=1}^{n+1} t_i > \tilde{\tau}_H, t_{n+1} \le \tilde{\tau}_I | A, B\} &= \Pr\{t_{n+1} \le \tilde{\tau}_I | A, B\} \cdot \Pr\{\sum_{i=1}^{n+1} t_i > \tilde{\tau}_H | t_{n+1} \le \tilde{\tau}_I, A, B\} \quad (40)\\ &= (1 - e^{-\lambda_f \tilde{\tau}_I}) \cdot \Pr\{M_f(\tilde{\tau}_H) < n+1 | M_f(\tilde{\tau}_H) \ge n\}. \quad (41)\end{aligned}$$

Plugging these into (39) yields

$$\Pr\{X_2 = n\} = (1 - e^{-\lambda_f \tilde{\tau}_I})^n e^{-\lambda_f \tilde{\tau}_I} \Pr\{M_f(\tilde{\tau}_H) \ge n\} + (1 - e^{-\lambda_f \tilde{\tau}_I})^{n+1} \Pr\{M_f(\tilde{\tau}_H) = n\}, \quad (42)$$

which can then be used to compute $\mathbb{E}[X_2]$.

Remark: While TTL cache with both a reset timer and a non-reset timer has been considered in [12], its characterization of the hit ratio is in terms of characteristics of the miss process, e.g., expected number of requests per miss (Lemma 1) and rates of generating misses (Lemma 4), which are unknown to begin with. Additionally, [12] did not consider access delays or multi-staged policies. As shown in (42), the existence of both timers causes $\mathbb{E}[X_2]$ and hence the hit ratio to have a non-closed form even under Poisson arrivals, which is in sharp contrast to the closed-form results in the presence of only one timer. We leave to future work the derivation of a TTL approximation under both timers that is easier to compute.

IV. Performance Under Cache Pollution Attack

We now apply the TTL approximation formulas obtained in Section III to analyze the performance of these policies under pollution 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

$$\sum_{f \in F_l} \frac{\lambda_f h^\pi(\lambda_f, T)}{\Lambda_l}, \quad (43)$$

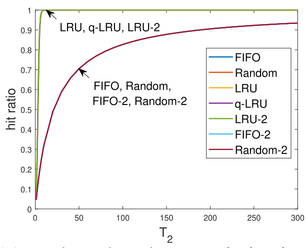
where T implicitly depends on the attack rates $(\lambda_f)_{f \in F_a}$ in addition to the legitimate flow rates $(\lambda_f)_{f \in F_l}$. We will focus on IRM in the rest of this section for explicit insights, although our approach is extensible to more general cases.

A. Optimal Attack Strategy

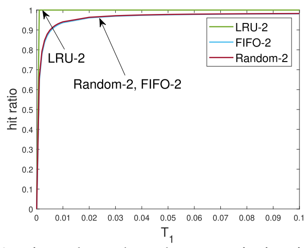
To understand the fundamental performance limit under pollution attacks, we first identify the optimal attack strategy against each policy. In particular, while a higher attack rate will always bring more damage, it also incurs a higher cost to the attacker. Therefore, the optimal attack strategy should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

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

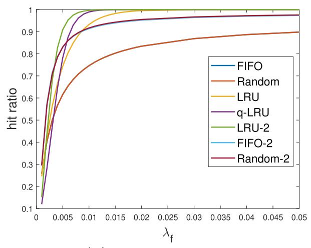
Theorem 1: If the characteristic time T is constant for all $(\lambda_f)_{f \in F_a} \in \mathcal{A}$, and $h^\pi(\lambda_f, T)$ is an increasing function of T



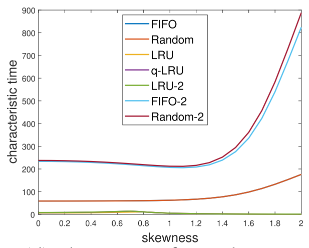
(a) real cache characteristic time



(b) virtual cache characteristic time



(c) request rate



(d) skewness of attack rates

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

and a concave function of λ_f, then the optimal attack strategy in $\mathcal{A}$ that minimizes (43) is to equally allocate the total attack rate, i.e., $\lambda_f = \Lambda_a/C_a$ for all $f \in F_a$.

Some of the assumptions in Theorem 1 are guaranteed to hold under certain conditions (see proof in Appendix.B).

Lemma 2: Under IRM, we have that:

1) $h^{\pi}(\lambda_f, T)$ for every policy π considered in Section III is increasing in T;
2) for $\pi =$ FIFO and Random, $h^{\pi}(\lambda_f, T)$ is concave in λ_f;
3) for $\pi =$ LRU and q-LRU, $h^{\pi}(\lambda_f, T)$ is concave in λ_f if $\bar{D}$ is sufficiently small and $e^{\lambda_f T} \geq (1-q)/q$.

While the remaining assumptions are not proved to hold exactly, we have verified numerically that they hold approximately for all the considered policies under IRM and rate allocations of interest. Specifically, while analyzing the concavity of the hit ratio wrt the request rate is intractable for the two-staged policies, we have verified the concavity numerically (Fig. 4 (c)). Moreover, while the characteristic time generally depends on the attack rate allocation, it remains largely constant for a wide range of skewness that corresponds to potentially good attack strategies (Fig. 4 (d)). Similar observations have been obtained 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 (43) as a function of C_a, which can then be minimized to choose the optimal C_a. However, as the characteristic time T is the solution to a high-order polynomial or transcendental equation that cannot be solved in closed form, (43) cannot be written as a closed-form function of C_a. Instead, we use other means to obtain insights, starting with the following observation (proved in Appendix.B).

Proposition 3: Under FIFO, Random, and LRU, $C_a = \infty$ is optimal in minimizing (43) under IRM.

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 3.

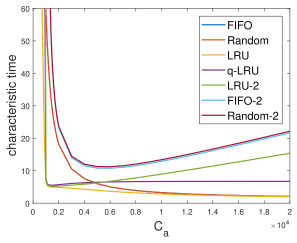


Fig. 5. Optimizing #attack flows (parameters in Section VI-A1, $\Lambda_a = 1000$).

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 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).

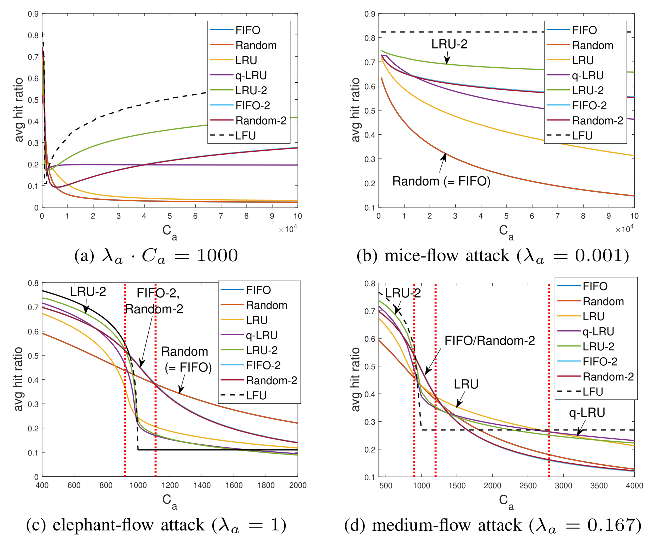
For example, under the setting in Fig. 5, the optimal C_a for q-LRU and LRU-2 is around 1000, which makes the rates of attack flows comparable to that of the largest legitimate flow, hence suggesting an elephant-flow attack; the optimal C_a for FIFO/Random-2 is around 6000, leading to much smaller attack 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 3, 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 (43); 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



(a) $\lambda_a \cdot C_a = 1000$ (b) mice-flow attack ($\lambda_a = 0.001$)

(c) elephant-flow attack ($\lambda_a = 1$) (d) medium-flow attack ($\lambda_a = 0.167$)

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

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 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 3, the behaviors of the other policies also have intuitive explanations. Specifically, we know that q-LRU and LRU-2 closely approximate LFU [7], 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; in contrast, simple indiscriminate policies (e.g., FIFO, Random) are resilient to elephant-flow attacks by treating all the flows equally, but vulnerable to mice-flow attacks as they tend to generate more attack flows. Under medium-flow attacks, LFU and its approximations 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 pollution 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:

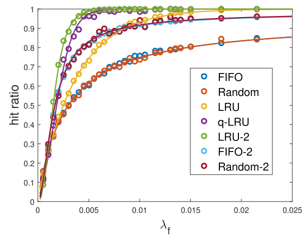
1) for each candidate policy $\pi \in \Pi$, solve the characteristic equation $\sum_{f\in F} h^{\pi}(\lambda_f, T) = C$ for the characteristic time T under policy π;
2) based on the calculated characteristic times, estimate the average hit ratio $\bar{h}^{\pi}$ of the legitimate flows under each $\pi \in \Pi$ by (43), where $F_l := F \setminus F_a$;
3) select the policy π^* with the maximum $\bar{h}^{\pi}$.

Remark 1: 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 [35].

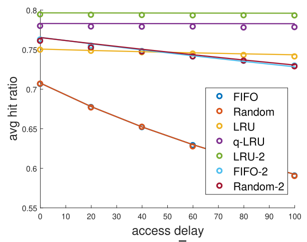
Remark 2: As TTL approximations only describe the cache performance in the steady state, the above policy selection scheme is only designed to maximize the (worst-case) average hit ratio of the legitimate flows in the steady state. We leave the design of new policies or policy selection schemes that accounts for transient behaviors to future work.

VI. Simulations

We evaluate the proposed solutions via simulations 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 pollution attacks due to its small size as shown in [9], [22]–[26]. 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.



(a) vs. λ_f ($\bar{D} = 20$)



(b) vs. $\bar{D}$

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

"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 [36], and the average access delay $\bar{D} = 20$ ms according to the performance of such switches [37]. 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 [38], Λ_l according to the average rate of the corresponding traces [39], 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 [39], 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 [38] 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, validating the correctness of our analysis. We further vary the average access delay and evaluate the average hit ratio over all the flows, as in Fig. 7 (b). Besides verifying the accuracy of our formulas under a wide range of access delays, this result also demonstrates the value of considering access delays, as ignoring such delays (i.e., assuming $\bar{D} = 0$) can cause significant overestimation of the hit ratios. Meanwhile, the result also indicates that access delays do not have much impact under LRU and its variations (q-LRU, LRU-2), intuitively because these policies allow popular contents to renew its TTL and stay in the cache for a relatively long period of time, mitigating the misses incurred during content access.

TABLE I
POLICY SELECTION FREQUENCY UNDER HYBRID ATTACK

	LRU	Q-LRU	LRU-2	FIFO-2	FIFO
mice	–	–	90	–	–
medium	1	5	77	–	7
elephant	–	–	–	86	4

2) Impact of Pollution Attack: Next, we evaluate the average hit ratio for legitimate users under pollution 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. For the traces, as the legitimate flows and their rates vary from trace to trace, we plot the distribution of average hit ratios over 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 that: (1) no single policy can optimize the performance for legitimate users in all the attack scenarios, and (2) the TTL approximations can guide us to the best policy in a given attack scenario.

3) Performance of Policy Selection: Since Fig. 9 (a–c) already validate that the TTL approximations will lead us to the best policy under static attacks, we now focus on time-varying attacks. To this end, we simulate a *hybrid attack*, 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 (b) for the second $1/3$ of the trace, and the elephant-flow attack in Fig. 9 (c) for the last $1/3$ of the trace. The intuition is to take advantage of the fact that none of the policies is resilient against all the three attack strategies. Fig. 9 (d) shows the distribution of the average hit ratios over all the traces for the adaptive policy selection scheme proposed in Section V ('Adaptive') as well as the individual policies, assuming that the attack parameters (Λ_a, C_a) are accurately estimated. We see that (i) the adaptive policy selection scheme achieves a better performance than any single policy under the hybrid attack, and (ii) two-staged policies are more robust than single-staged policies. Furthermore, we investigate the behavior of adaptive policy selection by examining the frequency (in terms of #times) of selecting various policies in each stage of the attack in Table I. The results confirm that the proposed scheme is able to combine the strengths of different policies by choosing the most resilient policy in each attack scenario most of the time.

The results in Fig. 9 (d) are based on the assumption that the attack parameters (Λ_a, C_a) are accurately and instantaneously estimated. While how to estimate these parameters is out of the scope of this work, we have conducted additional simulations to understand the sensitivity of the proposed policy selection scheme to estimation errors/delays; see Appendix.C.

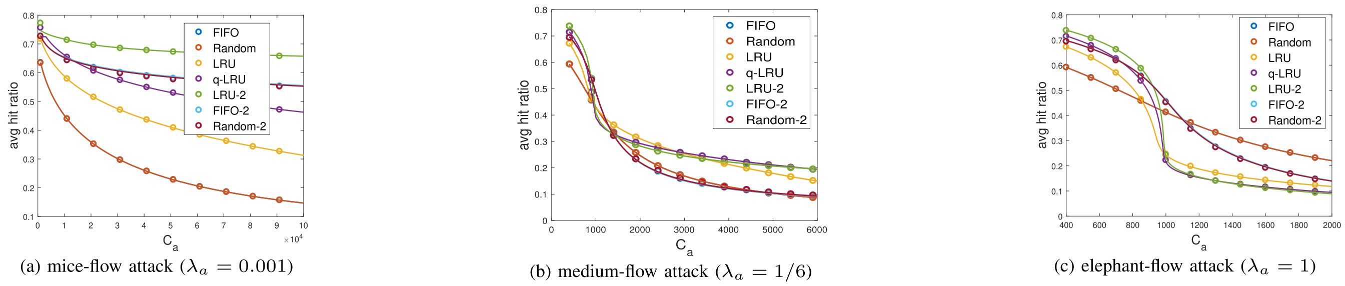

(a) mice-flow attack ($\lambda_a = 0.001$) (b) medium-flow attack ($\lambda_a = 1/6$) (c) elephant-flow attack ($\lambda_a = 1$)

Fig. 8. Pollution attack on synthetic traffic (○: simulated; —: predicted).

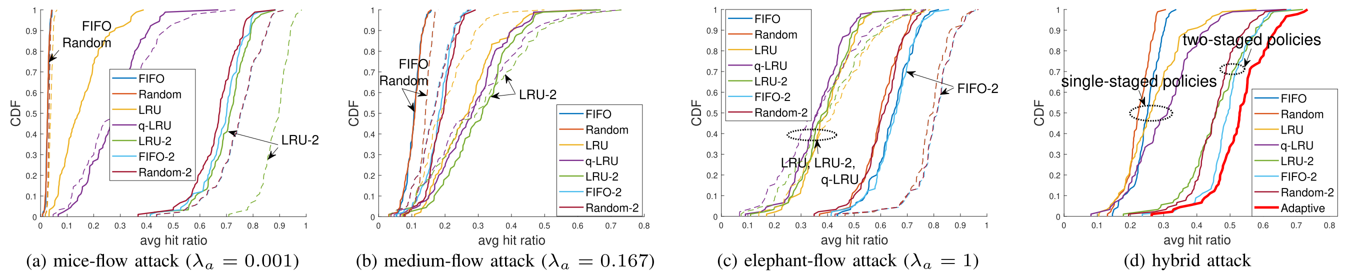

(a) mice-flow attack ($\lambda_a = 0.001$) (b) medium-flow attack ($\lambda_a = 0.167$) (c) elephant-flow attack ($\lambda_a = 1$) (d) hybrid attack

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

VII. Prototype Implementation and Experiments

We further implement selected policies in Open vSwitch, which is an open-source software-based SDN switch. Existing Open vSwitch implementation uses a fixed rule replacement policy that functions like LRU in the default setting [40]. Periodically, the eviction priority for all the non-permanent flow rules (those with idle or hard timeouts) is calculated, where the rule that expires the soonest has the highest priority. If a new rule needs to be inserted when the flow table is full, the rule with the highest eviction priority will be removed to make space for the new rule. As typically only the idle timeout is used, the above implementation leads to an approximation of LRU. We have also verified that Open vSwitch by default only uses one table (table 0), justifying treating the flow table as a single cache. As our previous result shows that a single policy will not be resilient to all attack scenarios, we implement additional rule replacement policies and test their resilience under various attack scenarios.

A. Implementation in Open vSwitch

1) Additional Rule Replacement Policies: In addition to the default LRU-like policy, we implement two new policies: FIFO and q-LRU. FIFO is implemented by changing the computation of eviction priorities (the earlier the creation time of a rule, the higher the priority for eviction), and q-LRU is implemented by inserting a coin flip in the function handling the insertion of new rules such that non-permanent rules are only inserted with probability q, which is a design parameter (set to 0.15 according to Section VI-A). Both are low-complexity stateless policies that impose a minimal overhead similar to LRU.

This implementation does not include Random and the two-staged policies. Random is skipped as its performance has been shown to be dominated by FIFO. While the two-staged policies have shown good resilience under some attack scenarios in the simulations, they have a much higher overhead (including memory consumption) due to the need to maintain the virtual cache. Moreover, we find that implementing these policies in Open vSwitch will require disruptive changes to the code (e.g., to extract a unique rule ID from both the uncompressed rule sent by the controller and the compressed version stored in the flow table). Therefore, we leave the implementation of such more complex policies to future work.

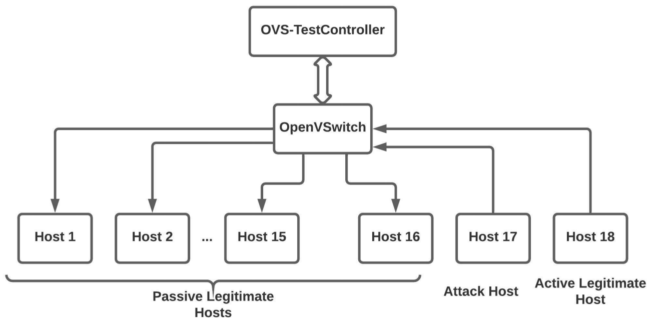
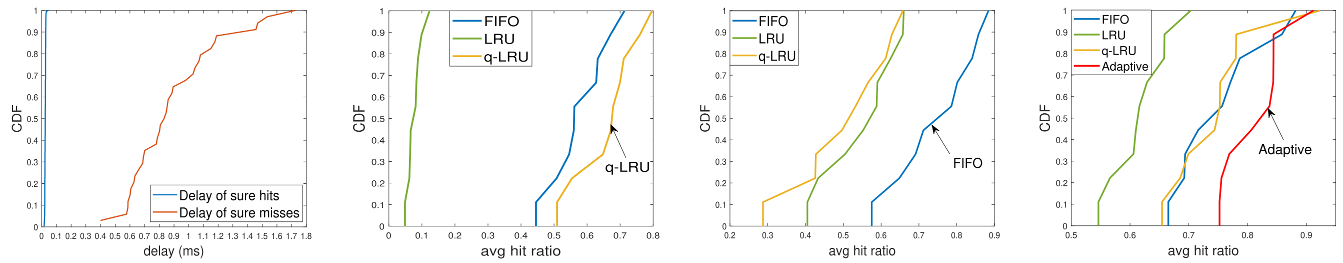


Fig. 10. Illustration of network created In mininet.

2) Mechanism for Runtime Policy Adaptation: Furthermore, to enable attack-aware policy adaptation [1], we implement a mechanism to track and adapt the replacement policy at runtime. This includes a new flow table attribute called `eviction_algorithm` to indicate which policy is selected, and a new command to configure this attribute. The new command utilizes the utility `ovs-ofctl` to generate a `table_mod` message that specifies the newly selected policy. This command is applicable to any switch supporting OpenFlow 1.4/1.5, which supports `table_mod`.



(a) hit/miss delays (b) mice-flow attack ($\lambda_a = 0.001$) (c) elephant-flow attack ($\lambda_a = 1$) (d) hybrid attack

Fig. 11. Experiment results in mininet: (a) CDF of hit/miss delays, (b)–(d) pollution attacks with total attack rate 1000 (unit: packets per 100 ms).

The modified code based on Open vSwitch v2.14.0 is available at [41]. We refer to [42] for more details.

B. Experimentation in Mininet

We test our implementation in Mininet [43], which is an SDN emulator that can emulate a virtual SDN based on the Linux kernel and the real network stack on a single machine. Our experiments are conducted in a VM with Ubuntu 16.04, 8 cores, 16 GB RAM, and 10GB hard disk.

1) Experiment Setup: We create a 18-host network with one attack host, one active legitimate host, and 16 passive legitimate hosts only receiving packets, all connected through an Open vSwtich under the control of an OpenFlow-Test Controller, as illustrated in Fig. 10. Using multiple receiving hosts allows us to create more flows, each defined by a combination of source and destination IP addresses, MAC addresses, and port numbers. The controller sets an idle timeout of 60s for each non-permanent rule (no hard timeout). We have verified that this timeout has negligible impact on our experiments.

Both the attack host and the active legitimate host generate UDP packets according to specified patterns. In each experiment, the attack host generates C_a flows according to independent Poisson processes of equal rate λ_a, where parameters (C_a, λ_a) are determined by the attack strategy. The active legitimate host generates flows according to one of the extracted traces in Section VI-A2. In both cases, different flows are generated by varying the destination as well as the port numbers. Each packet has a unique sequence number in the payload that allows us to compute the (one-way) delay between transmission and reception. We use the `usleep()` function to control the timing between packets according to the interarrival times in the Poisson processes or traces. The code and data for our experiments are available at [44].

2) Implementation Challenge: Our simulations use a total traffic rate of roughly 1000 packets/ms. However, generating packets at this rate in the experiment faces an implementation challenge: each call to `usleep()` incurs an overhead of 70–80 μs in our system, which implies a maximum rate of roughly 10 packets/ms. To accommodate this overhead, we send both the attack traffic and the legitimate traffic at 100x slowdown, i.e., changing the time unit from '1 ms' to '100 ms'. Due to the slowdown, the experiments are only performed on one trace from each of the 9 trace files from [39].

Ideally, we should increase the access delay $\bar{D}$ (i.e., time to obtain and install a new rule) proportionally. However, due to the limited buffer size at the interface between the switch and the controller, we can only impose up to 500 ms of (round-trip) delay without causing buffer overflow, which amounts to 5 ms without the slowdown. Such a rule installation delay is smaller than what can be achieved on commodity switches [37]. As we have observed that the performance gap between different policies increases with the access delay [42], we conjecture that suitably adapting the replacement policy can achieve an even greater performance improvement over the default policy in commodity switches than what is achieved in our experiments. Validating this conjecture will require modification to the internal logic of commodity switches, which is beyond the scope of this work.

3) Hit/Miss Inference: Following the approach in [9], we use the measured packet delays to infer whether a packet has incurred a hit or a miss at the flow table. To this end, we profile the delay distributions under sure hits and sure misses, *before adding any delay to the controller*, as illustrated in Fig. 11 (a). There is a clear gap between the largest delay for hits and the smallest delay for misses, which allows us to distinguish hits from misses. With the 500-ms additional miss delay, this gap will be even wider, allowing easy inference of hits/misses.

4) Experiment Results: Fig. 11 (b–d) shows the experiment results, where Fig. 11 (b) is under mice-flow attack, Fig. 11 (c) under elephant-flow attack, and Fig. 11 (d) under a hybrid attack which performs mice-flow attack for the first half of each trace and elephant-flow attack for the second half. Here we only consider mice-flow and elephant-flow attacks because they are the optimal strategies against the implemented policies: FIFO and LRU are most vulnerable to mice-flow attacks, and q-LRU is most vulnerable to elephant-flow attacks.

The experiment results confirm the value of using a replacement policy other than the default policy of LRU under attacks. Under the mice-flow attack, q-LRU substantially outperforms LRU; under the elephant-flow attack, FIFO substantially outperforms LRU. Moreover, under the hybrid attack, adapting the policy from q-LRU to FIFO according to the prediction by TTL approximation outperforms all the fixed policies.

VIII. Conclusion

Inspired by empirical studies that showed poor performance of normally good replacement policies under pollution attacks, we performed a systematic study of the attack resilience of a set of representative policies using the tool of TTL approximation. 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 with an SDN flow table as the cache validated our solutions, particularly that the flow table can be made more resilient to pollution attacks by suitably adapting the rule replacement policy based on the perceived level of attack. Our results also identified certain policies, especially FIFO-2, as an attack-oblivious solution with relatively good resilience. Our prototype implementation and experiments validated that our solution can improve the attack resilience of SDN switches.

References

[1] T. Xie, T. He, P. McDaniel, and N. Nambiar, "Attack resilience of cache replacement policies," in *Proc. IEEE INFOCOM*, May 2021, pp. 1–10.

[2] S. Podlipnig and L. Böszörmenyi, "A survey of web cache replacement strategies," *ACM Comput. Surv.*, vol. 35, no. 4, pp. 374–398, Dec. 2003, doi: 10.1145/954339.954341.

[3] S. Basu, A. Sundarrajan, J. Ghaderi, S. Shakkottai, and R. Sitaraman, "Adaptive TTL-based caching for content delivery," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1063–1077, Jun. 2018.

[4] I. Abdullahi, S. Arif, and S. Hassan, "Survey on caching approaches in information centric networking," *J. Netw. Comput. Appl.*, vol. 56, pp. 48–59, Oct. 2015.

[5] S. Alouf, N. Choungmo Fofack, and N. Nedkov, "Performance models for hierarchy of caches: Application to modern DNS caches," *Perform. Eval.*, vol. 97, pp. 57–82, Mar. 2016. [Online]. Available: https://hal.inria.fr/hal-01258189

[6] N. Gast and B. Van Houdt, "TTL approximations of the cache replacement algorithms LRU(m) and h-LRU," *Perform. Eval.*, vol. 117, pp. 33–57, Dec. 2017.

[7] M. Garetto, E. Leonardi, and V. Martina, "A unified approach to the performance analysis of caching systems," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 1, no. 3, pp. 1–28, May 2016.

[8] Y. Gao, L. Deng, A. Kuzmanovic, and Y. Chen, "Internet cache pollution attacks and countermeasures," in *Proc. IEEE Int. Conf. Netw. Protocols*, Nov. 2006, pp. 54–64.

[9] M. Yu, T. He, P. McDaniel, and Q. K. Burke, "Flow table security in SDN: Adversarial reconnaissance and intelligent attacks," in *Proc. IEEE INFOCOM*, Jul. 2020, pp. 1519–1528.

[10] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, Apr. 2004.

[11] P. Smith *et al.*, "Network resilience: A systematic approach," *IEEE Commun. Mag.*, vol. 49, no. 7, pp. 88–97, Jul. 2011.

[12] D. S. Berger, P. Gland, S. Singla, and F. Ciucu, "Exact analysis of TTL cache networks," *Perform. Eval.*, vol. 79, pp. 2–23, Sep. 2014.

[13] *Open vSwitch 2.14.90 Documentation*. Accessed: Sep. 2020. [Online]. Available: https://docs.openvswitch.org/en/latest/

[14] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1305–1314, Sep. 2002.

[15] 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?" *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 59–67, Jul. 2013, doi: 10.1145/2500098.2500106.

[16] B. Jiang, P. Nain, and D. Towsley, "On the convergence of the TTL approximation for an LRU cache under independent stationary request processes," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 3, no. 4, pp. 1–31, Sep. 2018.

[17] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance," in *Proc. 24th Int. Teletraffic Congr.*, 2012, pp. 1–8.

[18] A. Dabirmoghaddam, M. Dehghan, and J. J. Garcia-Luna-Aceves, "Characterizing interest aggregation in content-centric networks," in *Proc. IFIP Netw.*, May 2016, pp. 449–457.

[19] M. Dehghan, B. Jiang, A. Dabirmoghaddam, and D. Towsley, "On the analysis of caches with pending interest tables," in *Proc. ICN*, Sep. 2015, pp. 69–78.

[20] E. G. AbdAllah, H. S. Hassanein, and M. Zulkernine, "A survey of security attacks in information-centric networking," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1441–1454, 3rd Quart., 2015.

[21] Y. Zhou, K. Chen, J. Zhang, J. Leng, and Y. Tang, "Exploiting the vulnerability of flow table overflow in software-defined network: Attack model, evaluation, and defense," *Secur. Commun. Netw.*, vol. 2018, pp. 1–15, Jan. 2018.

[22] 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 *Proc. SECURECOMM*, 2017, pp. 356–376.

[23] Y. Qian, W. You, and K. Qian, "OpenFlow flow table overflow attacks and countermeasures," in *Proc. IEEE EuCNC*, Jun. 2016, pp. 205–209.

[24] B. Yuan, D. Zou, S. Yu, H. Jin, W. Qiang, and J. Shen, "Defending against flow table overloading attack in software-defined networks," *IEEE Trans. Services Comput.*, vol. 12, no. 2, pp. 231–246, Mar./Apr. 2019.

[25] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks," in *Proc. ACM CCS*, 2013, pp. 413–424.

[26] J. Weekes and S. Nagaraja, "Controlling your neighbour's bandwidth for fun and for profit," in *Proc. Secur. Protocols*, 2017, pp. 214–223.

[27] 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, pp. 174–188.

[28] C. Ghali, G. Tsudik, and E. Uzun, "Needle in a haystack: Mitigating content poisoning in named-data networking," in *Proc. SENT Workshop NDSS*, 2014.

[29] T. Kamimoto, K. Mori, S. Umeda, Y. Ohata, and H. Shigeno, "Cache protection method based on prefix hierarchy for content-oriented network," in *Proc. 13th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2016, pp. 417–422.

[30] 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 Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 447–456, Feb. 2014.

[31] H. Park, I. Widjaja, and H. Lee, "Detection of cache pollution attacks using randomness checks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2012, pp. 1096–1100.

[32] M. Xie, I. Widjaja, and H. Wang, "Enhancing cache robustness for content-centric networking," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2426–2434.

[33] N. Choungmo-Fofack, M. Dehghan, D. Towsley, M. Badov, and D. L. Goeckel, "On the performance of general cache networks," in *Proc. ValueTools*, Dec. 2014, pp. 106–113.

[34] N. Gast and B. Van Houdt, "Asymptotically exact TTL-approximations of the cache replacement algorithms LRU(m) and h-LRU," in *Proc. 28th Int. Teletraffic Congr. (ITC)*, vol. 1, Sep. 2016, pp. 157–165.

[35] Y. Fu, D. Li, S. Shen, Y. Zhang, and K. Chen, "Clustering-preserving network flow sketching," in *Proc. IEEE INFOCOM*, Jul. 2020, pp. 1309–1318.

[36] D. Kreutz, F. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[37] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "CacheFlow: Dependency-aware rule-caching for software-defined networks," in *Proc. SOSR*, 2016, pp. 1–12.

[38] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 267–280.

[39] T. Benson. *Data Set for IMC 2010 Data Center Measurement*. Accessed: Sep. 2019. [Online]. Available: http://pages.cs.wisc.edu/~tbenson/IMC10_Data.html

[40] M. Kerrisk. (Dec. 21, 2020). *Linux Manual Page*. [Online]. Available: https://man7.org/linux/man-pages/man5/ovs-vswitchd.conf.db.5.html

[41] T. Xie, N. Nambiar, and T. He. (2021). *Configurable Rule Replacement Policies in SDN: Implementation in Open vSwitch*. [Online]. Available: https://github.com/SophieCXT/SDN-Implementation-in-Open-vSwitch

[42] N. Nambiar, "Attack resilience of cache replacement policies: Implementation and experimentation in SDN," M.S. thesis, Dept. Comput. Sci. Eng., Univ. Park, State College, PA, USA, 2021. [Online]. Available: https://etda.libraries.psu.edu/catalog/18973nmn5265

[43] *Mininet*. Accessed: Sep. 2020. [Online]. Available: http://mininet.org/
[44] T. Xie, N. Nambiar, and T. He. (2021). *Attack Resilience of Cache Replacement Policies: Code and Data for Experiments in Mininet*. [Online]. Available: https://github.com/SophieCXT/Code-and-Data-for-Experiments-in-Mininet

Tian Xie (Student Member, IEEE) received the B.Eng. degree in software engineering from Wuhan University, China, in 2019. She is currently pursuing the Ph.D. degree in computer science and engineering with The Pennsylvania State University, University Park, PA, USA, in 2019. Advised by Prof. Ting He, her research interests lie in the area of networks security and networks modeling and joint optimization problems over cache networks. She was awarded as the “Top Ten Outstanding Luojia Scholars of the Year 2018” by Wuhan University.

Namitha Nambiar received the B.E. degree in electronics and communication engineering from the PES Institute of Technology, India, in 2016, and the M.S. degree in computer science and engineering from The Pennsylvania State University, State College, PA, USA, in 2021. She is currently working as a Software Engineer with the Graph Infrastructure Team, LinkedIn, Bay Area, CA, USA. Her research interests include distributed systems, computer networking, and distributed algorithms.

Ting He (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Cornell University in 2007. She is an Associate Professor with the School of EECS, The Pennsylvania State University. Her research interests include computer networking, performance evaluation, statistical inference, and machine learning. Her work received multiple Outstanding Contributor Awards from IBM, multiple awards from DAIS-ITA and NIS-ITA, and multiple paper awards such as the 2021 IEEE Communications Society Leonard G. Abraham Prize and the Best Paper Award at the 2013 ICDCS. She has served as the TPC Co-Chair for IEEE ICCCN in 2022, the Area TPC Chair for IEEE INFOCOM in 2021, and an Associate Editor for IEEE TRANSACTIONS ON COMMUNICATIONS from 2017 to 2020 and IEEE/ACM TRANSACTIONS ON NETWORKING from 2017 to 2023.

Patrick McDaniel (Fellow, IEEE) served as the Program Manager and the Lead Scientist for the Army Research Laboratory’s Cyber-Security Collaborative Research Alliance from 2013 to 2018. He is the William L. Weiss Professor of information and communications technology and the Director of the Institute for Networking and Security Research with the School of EECS, The Pennsylvania State University. His research focuses on a wide range of topics in computer and networks security and technical public policy. He is a Fellow of ACM and AAAS and the Director of the NSF Frontier Center for Trustworthy Machine Learning.