System Design Concept: Rate limiting

When you hear the word “rate limiting” your mind should bring up limitation, hindrance, or even exercising control on something. As the terminology suggests that is exactly what it means. You implement an algorithm into a system to control the number of API requests the system receives.

Rate limiting is the methodology adopted to limit the number of requests a user can make to an API endpoint. Large-scale distributed systems are often built with rate-limiting features to protect the underlying services and resources of the system from abusive attacks towards the system in the form of high-frequency API calls by bots.

Types of Attacks:

1. Brute force attacks

> *As the name suggests this is a brute force way of looking at every possible ways to attack the system to gather sensitive information. In this attack a malicious user tries to gather sensitive information such as passwords, credit card information and other sensitive and personal information.*

2. DoS and DDoS attacks

> *DoS and DDoS (Distributed Denial of Service Attacks) are targeted attacks on large-scale distributed systems by a vicious attacker with the help of a network of bots (commonly known as bot-nets) to disrupt the entire service platform of the company’s system.*

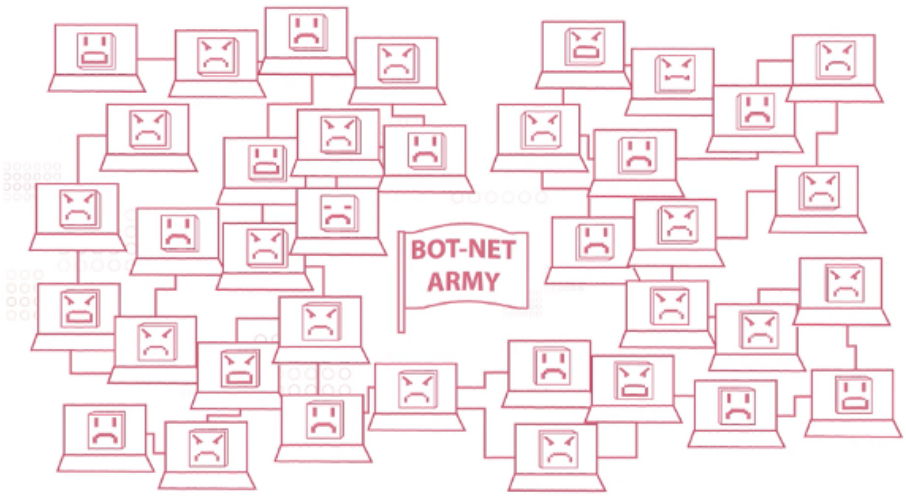

Image Source: Cloudflare | DDoS | What-is-a-DDoS-attack

3. Web Scraping

> *This is when a computer is programmed to extract out the valuable information from websites that have information exposed onto a publicly consumable API. The entire purpose of scraping algorithms is to scrape publicly available high value data.*

Why should we implement Rate Limiting?

Rate limiting allows you to improve the availability of the system such that all users requesting information from the system are supplied with data in an equitable manner.

No single user or a bot that throttles up the requests are provided with a response to expose the system to the bot such that it becomes unavailable for other users.

Secondly, rate limiting enhances the security of the system by preventing malicious attackers from destroying the system using bot-nets. Specifically, they use brute-

force algorithms to attack the system using a single machine or even multiple machines that constitute a network of bots known as a bot-net.

Different ways to rate limit

- *Token Bucket Algorithm:*

The token bucket algorithm enforces requests to be processed based on the token available for that request per unit time. In other words, when a request comes in, it is assigned a token from a bucket, such that the request is processed. If the request doesn't have a token to pick from the bucket, then this request is dropped. This limits the number of requests based on the number of tokens available for a period of unit time.

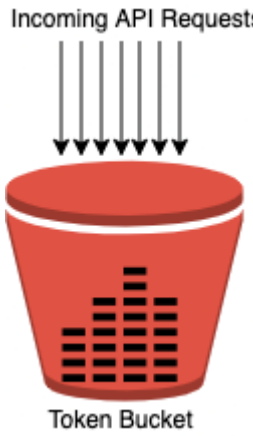

Image created by Author

- *Leaky Bucket Algorithm*

Alternative to the token bucket algorithm, a leaky bucket allows the user to send a burst of requests at a time that will be collected by some sort of a queue (or bucket). This queue processes requests at a consistent rate. Once the queue is full any more requests received from the user are leaked or dropped.

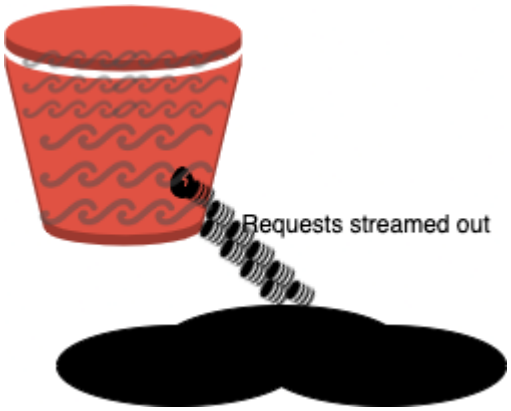

Image created by Author

- *Fixed Window Counter Algorithm*

A fixed window counter algorithm is another way to limit the requests based on a time frame window. A time window of size N allows for a stream of requests to be accepted by the system. However, the window has a threshold similar to the bucket algorithms we mentioned earlier. All requests that exceed the threshold within the same window are discarded.

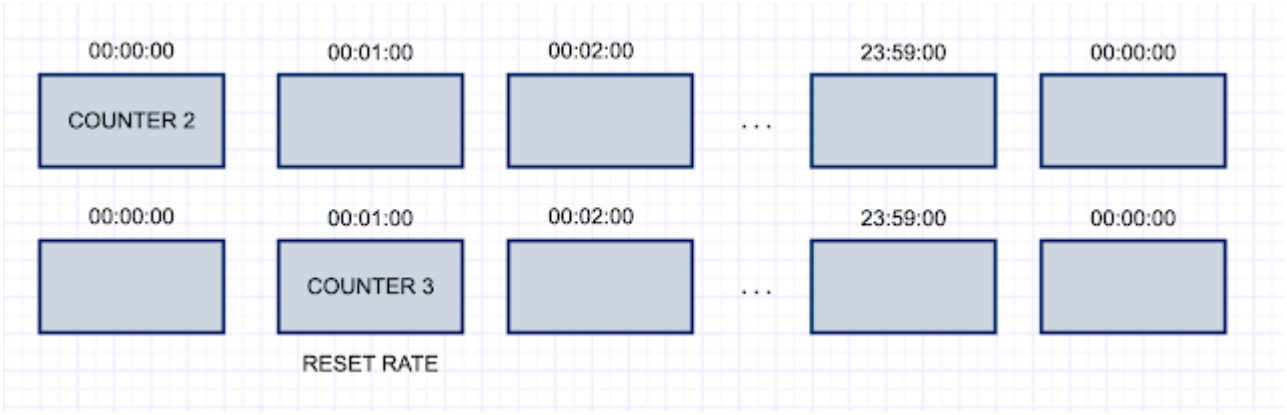

Image retrieved from Quinbay | Understanding Rate-limiting Algorithms

- *Sliding Log Algorithm*

This is a dynamic approach to recording the request rate for each fixed window by time stamping the requests, such that requests coming are ensured to not be dropped instantaneously. Instead, a log of timestamps is recorded to estimate the rate, and requests that exceed the calculated rate and held back for acceptance onto the next timestamp. This is pretty useful because it avoids the window being on a fixed timestamp.

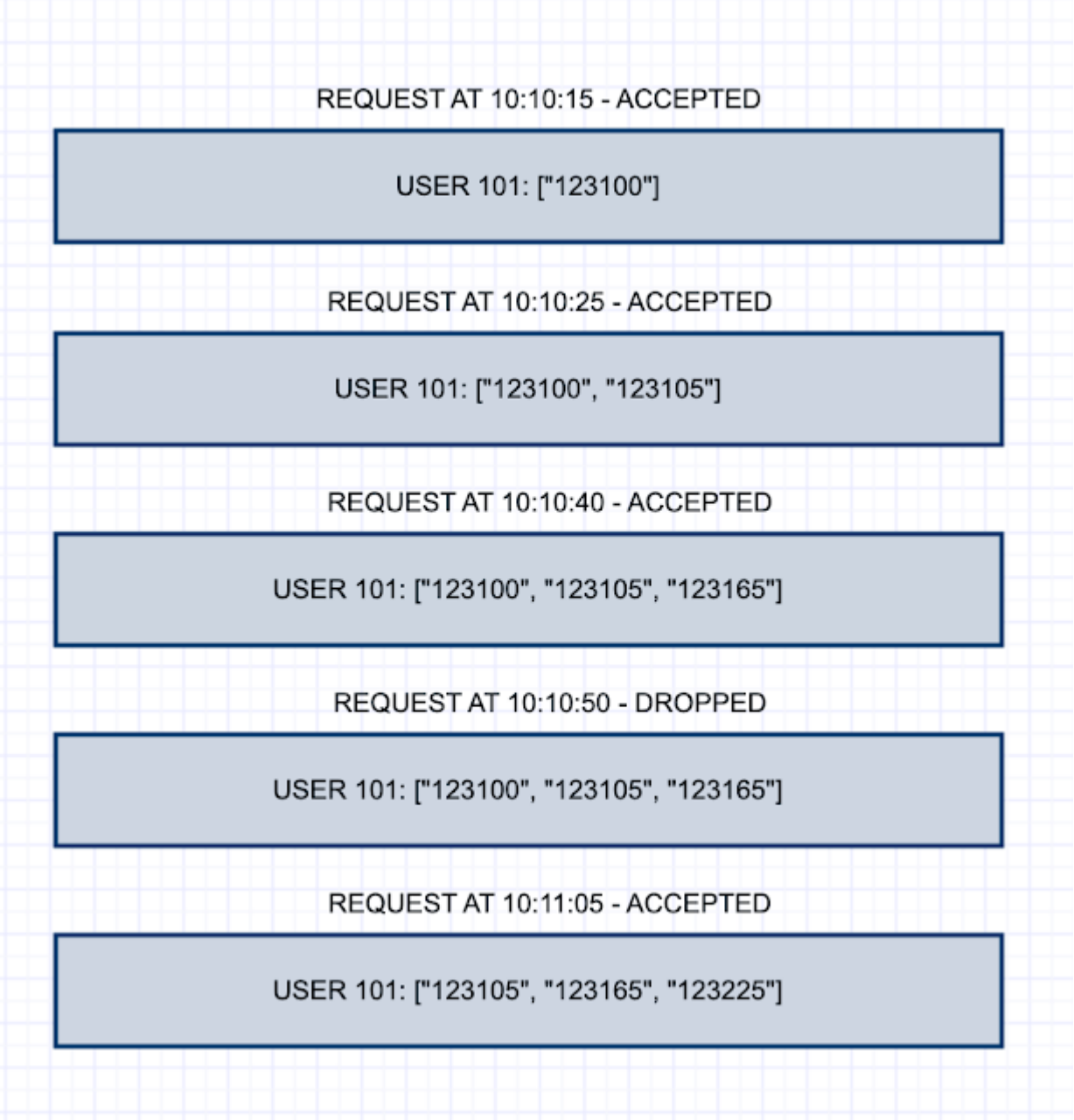

Image retrieved from Quinbay | Understanding Rate-limiting Algorithms

- *Sliding Window Algorithm*

This algorithm utilizes a hybrid approach of using the logging technique from sliding log and the window increments from fixed window to precisely calculate the rate of incoming requests. The constant and precise calculation of the rate allows for the smooth acceptance of the requests while simultaneously limiting bursts of requests in fractions of time.

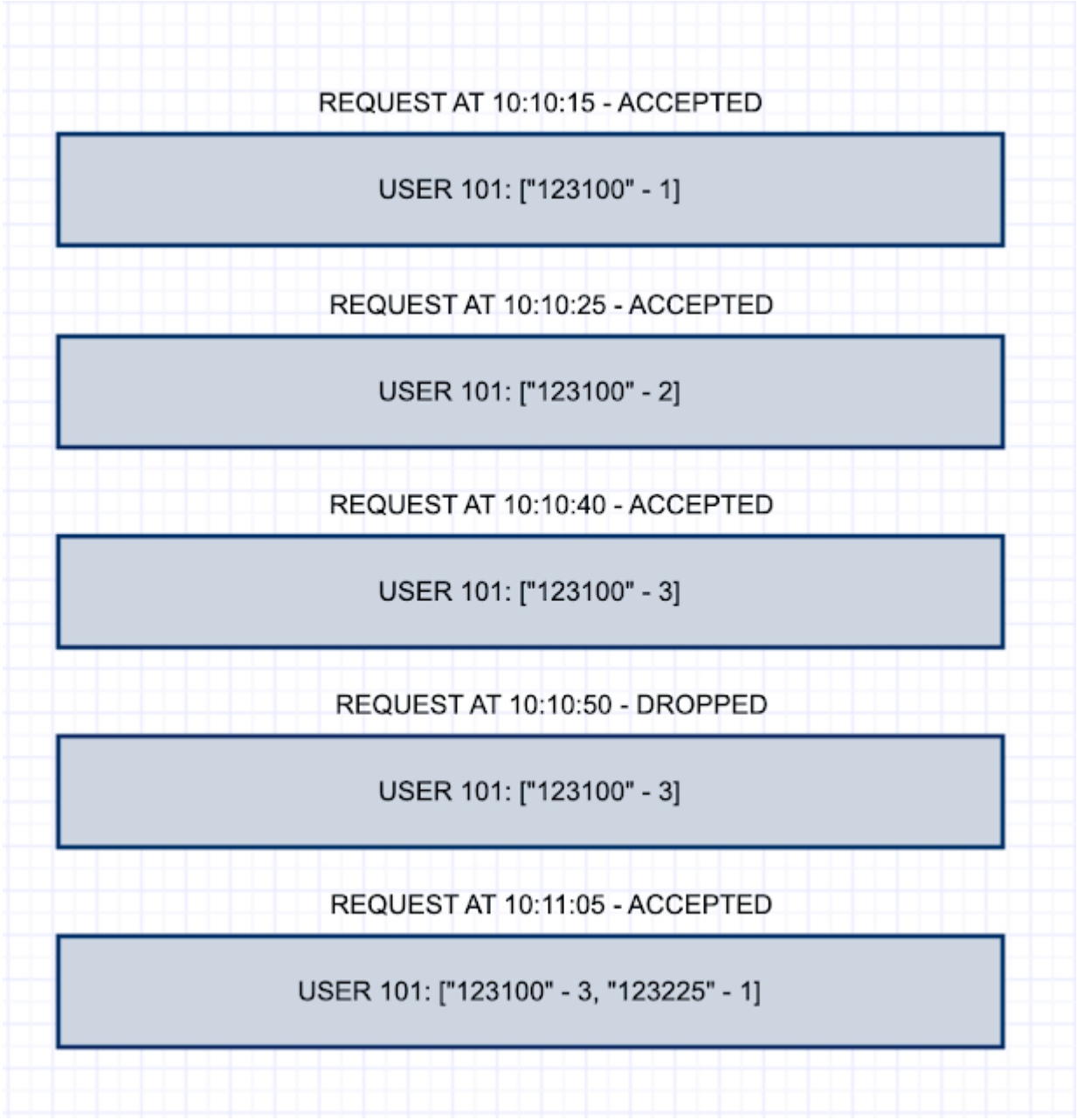

Image retrieved from Quinbay | Understanding Rate-limiting Algorithms

Best way rate limiting technique

My two cents on this section about the best ways to rate limit really depends on the system that you are building. While plowing through sources what I've learned is that there is no single best way to implement a rate-limiting technique. However, if your website/system is receiving large amounts of requests per second then the suggested technique for rate limiting would be Sliding Window Algorithm.

Unlike the leaky bucket algorithm where requests are starved and fixed window counter algorithm where denial of bursts of requests is avoided when you use the hybrid approach of the sliding window technique.

Conclusion

These rate-limiting algorithms are proven to work well for single-server systems. They provide a safe, effective and robust way to protect systems from malicious attacks of bot-nets or a vicious attacker. The novelty of the sliding window algorithm precisely calculates the rate while simultaneously maintaining the boundary conditions of a window of time, where the window of time is not fixed.

However, this only applies to requests coming into a single-server situation on a single location. What happens in the case of a distributed system with multiple nodes across different locations around the world. Are you going to implement multiple rate limiters on each of those nodes across the globe?

This is where things get slightly a bit more complicated. Subscribe and stay tuned for this on the next article drop.

> Thanks for your time! 🙂 and feel free to leave comments for feedback/discussions.
>
> One clap 👏 for the next system design article 🙂

Sources

[0] Xu, Alex. *System Design Interview — An Insider's Guide*. Independently published, 2020.

[1] "What Is Rate Limiting? | Rate Limiting and Bots | Cloudflare." *What Is Rate Limiting? | Rate Limiting and Bots*, Cloudflare, https://www.cloudflare.com/learning/bots/what-is-rate-limiting/.

[2] Singh, A. (2022, January 4). *System Design Basics: Rate Limiter — Geek Culture*. Medium. https://medium.com/geekculture/system-design-basics-rate-limiter-351c09a57d14

[3] Srikantaiah, N. (2020, December 28). *Understanding Rate Limiting Algorithms*. Https://Www.Quinbay.Com/. https://www.quinbay.com/blog/understanding-rate-limiting-algorithms