International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Token Bucket: Architecting Next-Gen Secure, Resilient, and High-Performance Distributed Systems and App Modernization

Nitin Gupta

Submitted: 10/01/2025 **Revised:** 24/02/2025 **Accepted:** 16/03/2025

Abstract*:* With the increase demand of digital footprint, creating a robust and resilient distributed system adhering to cloud needs has become essential. Distributed system has challenges of scaling and cascade failure in case of exponential or burst traffic. Token Bucket algorithm is an architecture paradigm in distributed system that not only helps to overcome these challenges but also to sustain increasing workloads. In digital eco-system Token Bucket helps systems to be loosely coupled with asynchronous processing that serves to design a system with maximum availability and minimum latency. Along with its self-healing and fault tolerance capability Token Bucket also helps in safeguarding the system from various cyber threats. Token bucket provides a blueprint to architect complex software system that delivers speed, reliability and fortified protection. This algorithm when properly designed and considered for use case can be a state-of-the-art for digital ecosystem.

Keywords: *Token Bucket Algorithm, Distributed Systems, Resiliency, Architectural Pattern, Fintech, Cloud Native*

1. Introduction

In today's modern digital systems, the demand for IT workload has increased exponentially, to meet this increasing demand the need for robust infrastructure is more significant. This modern distributed application spanning across various industries from Finance to Healthcare not only needs to be functional and robust but also needs to be more resilient and secure as they support day to day critical operations in human life [1]. Traditional digital system lags the capability to scale beyond the limit and often struggle to operate on these gigantic data loads, so the need for disturbed interconnected system with their own scaling capability which can proactively support these challenges in increasingly desirable.
This paper introduces an architecture paradigm which helps to cater these challenges of distributed system while applying Token Bucket Algorithm. Earlier Token Bucket Algorithm was known for its role in controlling the network traffic and network packet, but this work demonstrates how it can be applied to the modern digital infrastructure to meet the rigorous demand of increasing workloads [2]. By changing the context and applying the principles in next-generation applications, this design is helping system to be more secure, resilient and superior performance.
The Token Bucket architectural paradigm provides an advance design pattern that emphasize on loose coupling, distributed principles and asynchronous communication. This design pattern is crucial for maximum throughput and minimum latency in distributed system. Token Bucket design pattern helps to safeguard the infrastructure from cyber threats by prioritizing the request and increasing system availability. Its self-healing and resilient capability outshine in case of system disturbance or failures.

[1] *Nitin Gupta, Workday Inc, Dublin, USA*
ORCID ID : 0009-0002-1816-468X
** Corresponding Author Email: nitingupta.sri7@gmail.com*

The design patterns help to inherit robust control features and isolation mechanisms with loose coupling.
Token Bucket architecture leverages core features [3], to design sophisticated software system for distributed digital world. It helps to deliver high available, fault tolerant, reliable and secure digital system that can be relied for critical operations. This paper outlines the different use cases, when this pattern is applied with careful and thoughtful thinking can deliver a state-of-the-art solution for enhancing reliability, availability, security and resiliency of digital eco-system.

2. Architecture and Design

The fundamental architecture and design of this pattern is derived from the name of the algorithm The Token and The Bucket. These two elements are not for the namesake but plays a vital role in overall design. The design resonates with digital infrastructure needs for high availability, resiliency and security. It extends the traditional Token Bucket algorithm to align with modern next generation digital software to achieve robust and secure systems.
The Token represent the unit of work or an entity like a Gate Pass while The Bucket is a reservoir which holds the token. The bucket capacity governs how many Passes can be accumulated for any given time and act as a buffer for controlling the burst of operations or activity that can pass during that time [4]. Each operation or activity needs a token to do its works so by rate limiting the tokens in the bucket number of operations can be controlled without causing any disturbance to the application.
Applying the Token Bucket algorithm in modern system aligns with distributed system architecture principles like loose coupling, self-operability, CAP theorem. This design introduces standardized and performant system of managing resource consumption, access and flow across different system without compromising any of the design principles of distributes system. It allows system inter-operability without introducing any constraints

or compatibility issues.

2.1. Other Recommendations

The flexibility and control introduced by the design help to address various critical architecture concerns like [5]:
Rate Control – To control request rate from one system to another.
Access Control – Limiting the access control from the system.
Throttling - Preventing the system overloading.
Detection and Governance – Determining the priority of request and action on it.

2.2. Key Parameters

The token bucket algorithm is mainly governed around few key parameters which dictates it. These are:
Token – It defines the permission for unit of work Bullet.
Token Rate – It define the rate at which tokens are added.
Bucket Size – The maximum capacity of tokens a bucket can hold.
Burst Size – This determines number of tokens can be consumed at a once.

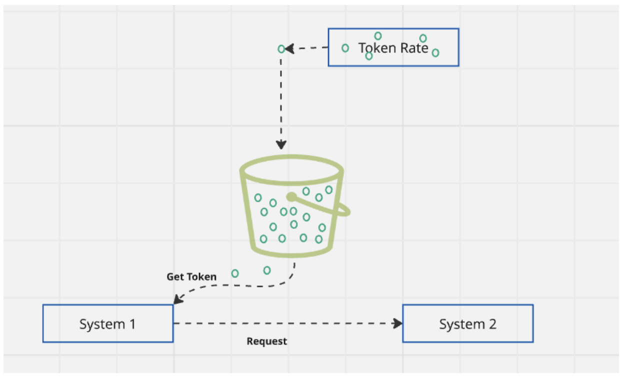

Fig. 1. Token Bucket Algorithm.

3. Analogy

This architecture pattern is analogous to a retail store handling influx of customer during holiday season sale. During holiday season large crowd wants to enter a favorite brand store to take the advantage of discount on their products, so before a store opens, they form a queue outside a store. Since store has limited capacity, they can allow only few customers to enter at a time which can be served by limited number of salespersons, so they distribute gate pass or coupon to enter the store. When the store initially opens number of customer equivalent to maximum capacity is allowed to enter the store, this is like a burst traffic on digital system whose capacity is managed by total number of tokens in the bucket. This demonstrates the ability of absorb the immediate surge before settling to a more controlled flow by continuous token distribution. As the shoppers completes the shopping and exits the store more token as given to the people in the queue so they can enter the store. This token distribution at a constant rate not only helps to allow only limited people inside the store but also to have a great customer experience as store associates are able to serve them better. This approach not only manages the peak demand but also helps to maintain store integrity.

4. Methodology

The token bucket algorithm is well suited for distributed architecture software applications in various industries where resiliency, performance and security play a vital role. By using inherit characteristics of this algorithm system availability can be maintained without affecting user experience or without comprising the application security. In this pattern token represents a permitted request which are added at a fixed rate to bucket. A request consumes the token from the bucket and if token is not available request is either rejected, or it will wait till token is available (this is per use case need). In case of low traffic token get accumulated in the bucket which can be consumed during burst load. As there is a flexibility of token rate control by altering it in case of issues system can be self-healed without changing infrastructure components.
The implementation of bucket varies as per business needs as in distributed system if token need to be shared by multiple services, then database can be used to store the token and to increase the performance shared cache e.g. Redis cache can sit on top of it, as database operations are costly [6]. Similarly token generation system can be job or API call from a scheduler.

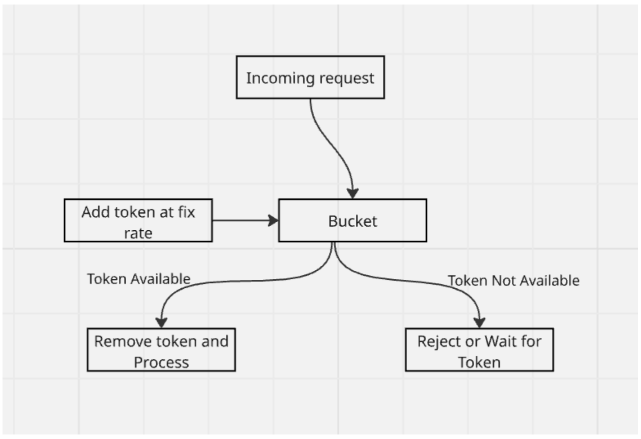

Fig. 2. Token Bucket Flow.

4.1. API Rate Control

Let's consider a scenario of holiday season when people are shopping more and it is common for the users to check their bank account or check recent transactions. This increase in traffic can easily overload the downstream systems and slow the API response or can lead to system failure which will impact the whole ecosystem. This can also lead to a cascade failure to multiple interconnected system and cause widespread disturbance.
Token Bucket Algorithm plays a critical role here by controlling the traffic to downstream system by updating token at a constant rate so downstream system in not overwhelmed and continue to serve as is. Here is how it works:
Each request to check account or transaction need a token to proceed further.
The tokens are added at a constant rate to meet the system capacity and not to overwhelm it.
The bucket is defined with maximum capacity of tokens it can hold. It's like a burst traffic when users first start to check at the beginning of the sale.
If bucket runs out of the token further requests are either put in the queue or throttled based on the design to have better customer experience.
This approach helps the system to handle the burst traffic which never exceeds the system capacity and continue to serve without failure or latency. This result in high available and resilient infrastructure in Banking and Fintech domain that can be trusted with increasing workloads.

4.2. Security and Fraud Detection

With the increase in digital footprint of software infrastructure the cyber threats to these systems have also increased hand in hand. The exponentially growth in AI space has led to escalate the sophisticated threat to digital ecosystem, thus need to secure and detect fraud to these systems has become overwhelmed.

Token Bucket algorithm fits perfectly to this architecture paradigm as fraud detection to secure an infrastructure is critical and crucial. Any lapse in it can lead to financial loss, reputation damage or data compromise:

Intelligent Traffic Management**:** Implementing a Token Bucket algorithm at the entry point of fraud detection API can help enterprise to have precise control for incoming request. The prioritization of the checks from different system can be done by this, the checks from high-risk transaction are given high priority and allowed to pass first, such system can have higher rate of token refill and can allow more burst traffic if needed so these transactions are always first to be checked.

Controlled Burst Tolerance: During the high traffic scenarios there be ca be case when the need to check to fraud detection is more. In such cases bucket capacity enables these checks to be requested and processed so no fraud activities succeed.

Enhanced System Stability and Performance: By controlling the flow Token Bucket prevents fraud API and system around it to be overwhelmed and safeguard them from becoming unstable. This process serves as a throttler which helps the whole ecosystem to be resilient and available during peak load hours.

This approach helps the system to be more secure during peak load hours without compromising the stability or availability. The system performs well to detect fraud even when it is tried to be overloaded to break or bypass with requests.

4.3. Microservices Inter Communication

In a distributed microservices environment inter communication between different services is essentials and lead to the success of application by completing transaction in different phases. The transaction divided into small unit of work which each service performs to complete the whole transaction [7]. This modularity is great for independent scalability and agility but it brings the challenge to manage the flow of requests between the services so it should not become bottleneck or collapse in high load [8].

Token Bucket algorithm comes handy solution to manage and control this whole ecosystem during burst or high traffic scenarios. To control any high number of request greater the system capacity bucket helps to manage it so only requests that can be managed and services will proceed. One service can receive a significant traffic but if it propagates that traffic to other system which is not capable to handle such loads, it can be overloaded, slowed or result in failure. This failure or latency in system can cause cascade effects to other services and result in failure of whole application. So, token bucket helps here by:

Maximum rate at which one service can send requests to another.

The burst of requests that can be handled of services.

If a service Reduce domino effect in whole system.

It can help to circuit break in the service, so request is backed up till token is available in the bucket.

Control the communication with third party system, so their delay or slowness does not result failure or latency.

5. Evaluation and Key Considerations

While the token bucket algorithm is an excellent fit for many applications modernization use cases, by effective throttling request and enhancing high availability, it's crucial to acknowledge it does not fit same with parameters in all the solution. There are key considerations that needs to be evaluated before it needs to be implemented in any use case and tune as per the user case customization.

Without evaluation of this consideration can cause significant issues in the system. If these evaluations are not done properly, it can hamper the system performance and increase the latency which can become bottleneck in overall system throughput. Therefore, thorough understanding of its specific context is successful for the integration.

Table 1. Evaluation and Considerations

Evaluation	*Consideration*
Traffic Handling	Allows bust traffic by saving tokens in bucket so need to think what bucket size suits. Rate limit the token creation provide steady flow of traffic.
Flexibility	It provides a flexibility by changing the token generation rate so as per need it can be changed.
Buffering	It provides data buffering till token are available.
Bucket Size	Appropriate bucket size should be chosen otherwise it can lead very big burst size.

6. Conclusion

Modern digital infrastructures need to meet increasing demands being robust, efficient, secure and highly available. This paper introduces the concept of Token Bucket algorithm the architectural paradigm which helps to meet demands and desired feature in modern digital systems. It demonstrates by leveraging the concept of Toke and Bucket provides effective mechanism for architecting next-generation systems.

It emphasizes on distributed design principles, asynchronous communication and isolation mechanism, The token bucket paradigm offers a blueprint to manage complex software systems. It provides the capability to regulate traffic flow, manage burst and prioritize requests to ensure optimal throughput and minimal latency. The design shields for fault tolerance with self-healing capability, thereby enhancing overall system availability and mitigating the risk of cascade failures across interconnected service [9].

As illustrated through practical use cases like API rate control in banking, Fraud detection in Fintech and inter service communication in distributed system architecture, The token buckets algorithm proves to be great player in diverse domains and use cases. It not only helps applications to safeguard the infrastructure but helps to secure data and provide high availability without compromising security.

Though the implementation of token bucket algorithm requires some key considerations like token rate, bucket size and shared cache strategies in distributed system but it brings lot of architectural benefits with it. When it is designed thoughtfully by considering the context this architecture creates an art-of-the-system solution for enhancing stability, security, resiliency and availability for a complex and distributed digital ecosystem.

Conflicts of interest

The authors declare no conflicts of interest.

References.

[1] N. U. Ahmed, Qun Wang, L. Orozco Barbosa, Systems approach to modeling the Token Bucket algorithm in computer networks, Wiley, Aug 2002, https://doi.org/10.1080/10241230215282.

[2] Puqi Perry Tang, T.-Y.C. Tai, IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320), March 1999.

[3] Peter L Dordal, Loyola University Chicago, An Introduction to Computer Networks, July 2020, intronetworks.cs.luc.edu.

[4] Salman A. Alqahtani, – Wireless Personal Communications: An International Journal, Volume 84, Issue 2, Pages 801 - 819, https://doi.org/10.1007/s11277-015-2661.

[5] Liu, W., et al. (2019). – A distributed cache consistency maintenance mechanism based on redis cluster. Journal of Computer and Communications, 61.

[6] S. Weerasinghe and I. Perera, "Optimised Strategy for Inter-Service Communication in Microservices," Int. J. Adv. Comput. Sci. Appl., vol. 14, no. 2, 2023, doi: 10.14569/IJACSA.2023.0140233.

[7] V. Lenarduzzi, F. Lomio, N. Saarimäki, and D. Taibi, "Doesmigrating a monolithic system to microservices decrease the technical debt? " J. Syst. Softw., vol. 169, p. 110710, Nov. 2020, doi:10.1016 /j.jss. 2020.110710.

[8] Jacopo Soldani, Giuseppe Montesano & Antonio Brogi, Department of Computer Science, University of Pisa, Pisa, ItalyWhat Went Wrong? – Explaining Cascading Failures in Microservice-Based Applications Conference paper, First Online: 26 September 2021, pp 133–153.

[9] David Gamez, Simin Nadjm-Tehrani, John Bigham, Claudio Balducelli, Kalle Burbeck, – Tobias Chyssler, Safeguarding Critical Infrastructures, 2017 - opendocs.ids.ac.uk.