Token Bucket

Contents

Authors:
Praveen Kumar Pendyala - 2919474
Lokesh Kumar Jamjoor Ramachandran - 2596208
Felix Euteneuer - 1909362

1. Introduction

In the bonus system for the lecture “Communication Networks I”, we as a group have chosen to implement **Token Bucket** algorithm.

The **Token Bucket**[3] is an algorithm used in packet switched computer networks and telecommunication networks. It can be used to check that data transmission, in the form of packets, conform to defined limits on bandwidth and burstiness.

A **Token Bucket** consists of a bucket that can hold up to **b** tokens. Tokens are generated at a constant rate of **r** tokens per second and added to the bucket as long as it is not filled completely. Each packet transmitted into the network must first remove a token from the token bucket. If the bucket is empty, the packet must wait for a new token or alternatively it is dropped or marked with a lower priority. A token bucket is able to police a bursty data stream with the average rate r and a maximum burst size of b packets.

2. Algorithm

The Token Bucket algorithm has been represented below in a simple way. The algorithm is as follows:

- A token is added to the bucket every **1/r** seconds.
- The bucket can hold at the most **b** tokens. If a token arrives when the bucket is full, it is discarded.
- When a packet of **n** bytes arrives, **n** tokens are removed from the bucket, and the packet is sent to the network.
- If fewer than **n** tokens are available, no tokens are removed from the bucket, and the packet is considered to be non-conformant.

3. Realization

For the implementation of a visual representation of the algorithm, we chose Web as our platform. The web app has been tested to be working on all major web browsers such as Firefox, Google Chrome and Safari.

3.1 Libraries and Tools

The entire project is realized in Webstorm[4] IDE using an Academic license. The following list of libraries are used in the implementation.

1. Bootstrap[5]
2. GreenSock GASP[6]
    a. TweenLite[7]
    b. TimelineLite[8]

All the libraries are referenced and fetched from CDN servers and the application runs entirely on client's browser.

1. *Bootstrap*

    Bootstrap is a free and open source collection of tools to create websites and web applications. It contains HTML and CSS based design templates for topography, forms, buttons, navigation and other interface components, as well as optional Javascript extensions. JQuery is added as part of the Bootstrap Library.

    Bootstrap wraps the whole page in container and uses the concept of Rows and Columns to simplify layout designs that can be supported across various screen sizes. In Bootstrap, each row is divided into 12 columns.

2. *GreenSock GASP*

   The GreenSock Animation Platform is a suite of tools for scripted animation. GreenSock offers a wide range of plugins and core components. We used two GreenSock components - TweenLite and TimelineLite, in our implementation.

   a. **TweenLite :** the core of the engine which handles animating just about any property of any object. It is relatively lightweight yet full-featured and can be expanded using optional plugins. For instance, we are using the CSSPlugin for animating DOM element styles in the browser.

   b. **TimelineLite :** a powerful, lightweight sequencing tool that acts like a container for tweens, making it simple to control them as a whole and precisely manage their timing in relation to each other. You can even nest timelines inside other timelines as deeply as you want. This allows you to modularize your animation workflow easily.

3.2 Implementation overview

The entire application code is spanned across 3 files:

1. index.html
2. app.css
3. app.js

A brief overview of the contents of each file is given below.

1. *index.html*

   This is the visual skeleton of our web application. Building on top of the Bootstrap framework, our application is divided into 4 Rows.

   i. Row 1
   Page title and Github Star

   ii. Row 2
   Controls for Packet Rate and Token Rate along with Token Bucket container. Within the Token Bucket, The tokens are spanned across 4 rows with 5 tokens per row.

   iii. Row 3
   Packet Source, Traffic Tunnel, Discard bin and Destination network

2. *app.css*

   The styling of various elements in the app. The application is built **without** using any images. Every element is designed with the help of CSS styling.

3. *app.js*

   The backend logic to realize the Token Bucket Algorithm. All the necessary element models and methods for animating various elements animation. More details on contents of this file are covered in the next section.

3.3 Implementation specifics

This section discusses the fine details of the backend logic for Token bucket realization and animation of various elements in our application.

There are 4 major aspects of the implementation.

1. Controls
2. Models
3. Generators
4. Animations

The starting point of the application backend is the **init()** function which is called in Window.onLoad()

1. *Controls*

   To control the rates of Token generation or Packet generation. There are four functions in total.

   i. increasePacketRate()
   ii. decreasePacketRate()
   iii. increaseTokenRate()
   iv. decreaseTokenRate**()**

2. *Models*

   There are 3 models in total. Each of them represent the backend of a specific type of element.

i. Packet()
ii. Token()
iii. TokenAnimator()

The use of Packet and Token are trivial from their names. The third model TokenAnimator is used to represent a special Token which is used for all Token Animations.

3. *Generators*

Two different generators are implemented.

i. TokenGenerator()
ii. PacketGenerator()

4. *Animators*

Theses are methods, in each of the Model objects, that perform the animation tasks. Each object has it's own timeline, denoted by **tl,** on which it animates. Thus all elements in the application are animated independently with no interference on each other. Animators are defined only for Packets and TokenAnimator.

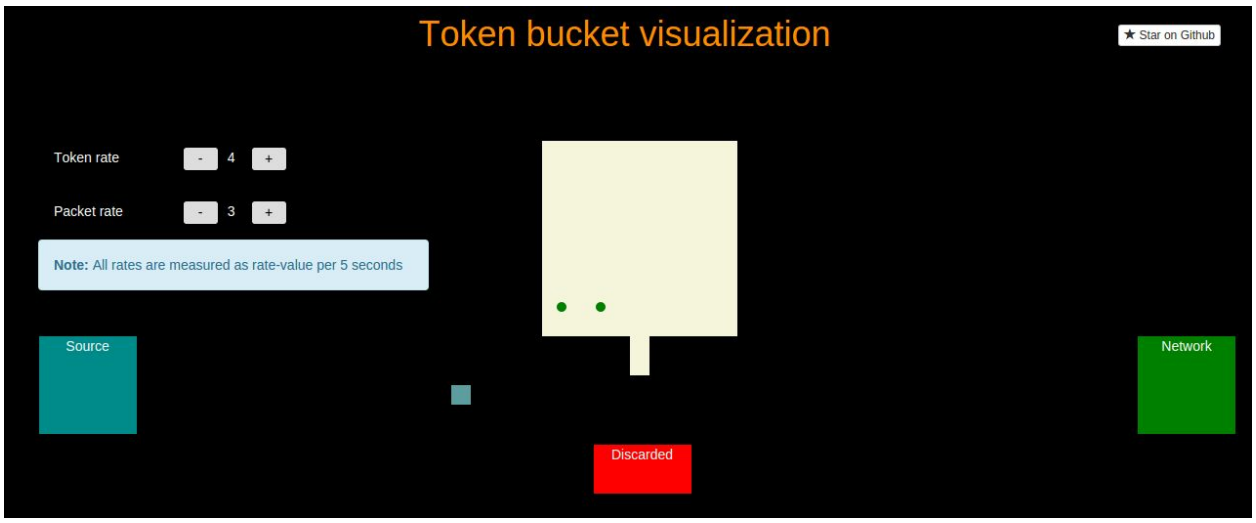
i. Packets - If Tokens are available, Packets can be animated to the network by adding a token to it. If Tokens are not available, then packets are animated to the discard bin below the bucket.

ii. TokenAnimator - Presents an animation of a Token being added to the Packet arrived at the bucket.

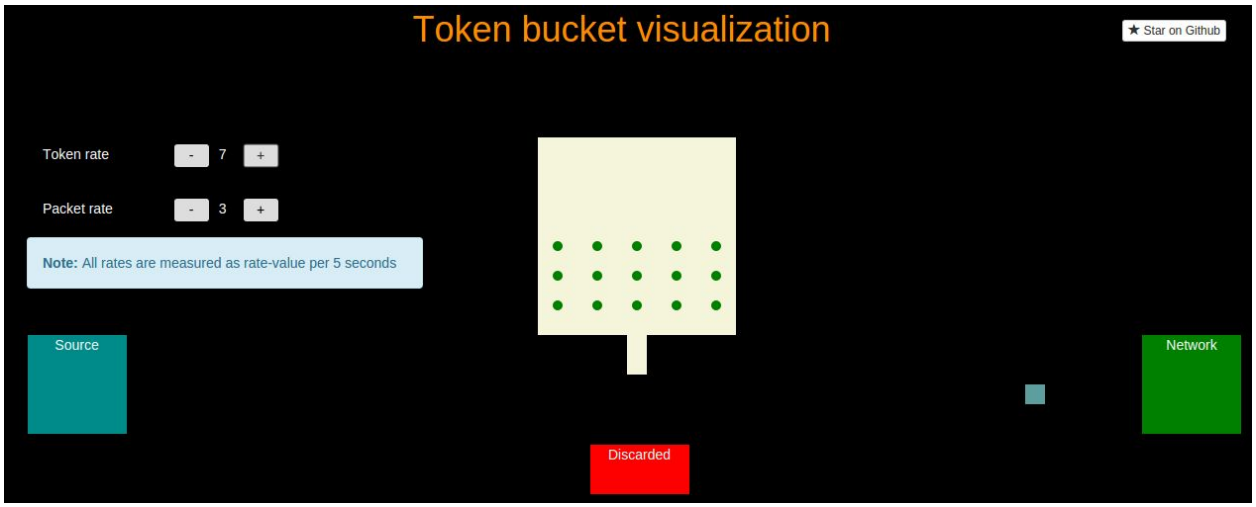
More details of the implementation can be referred from source code[1] on Github

4. Demo

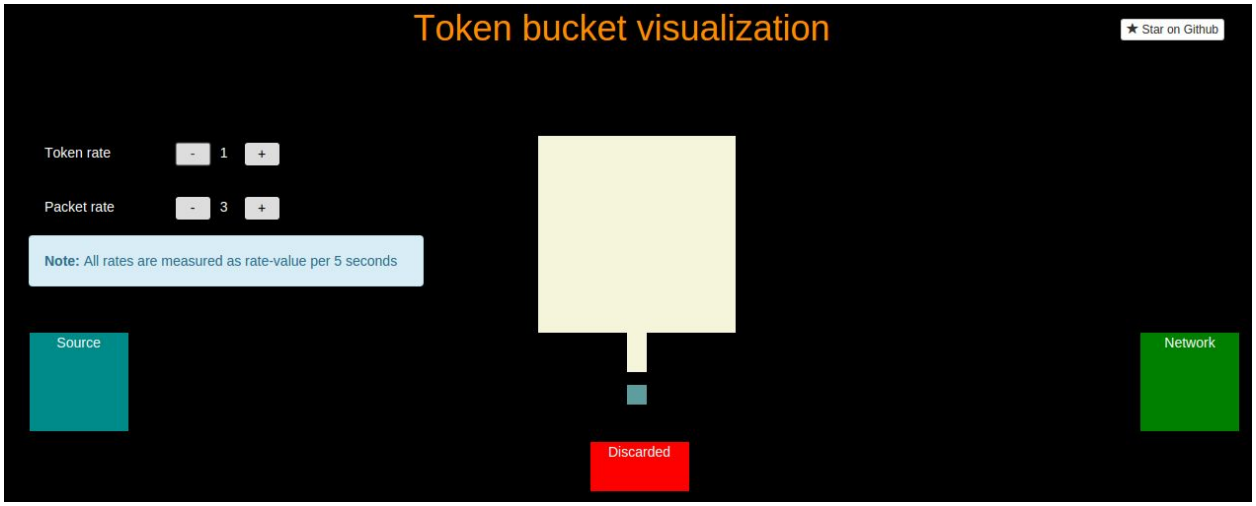
A couple of screenshots of the application are shown below. Please note that a live demo can be accessed at http://praveen.xyz/demos/token-bucket/ [2]



Packet reaching the Token bucket from source



Packet received a token so, it can enter the destination network



No tokens left in the bucket. Packet will be discarded to the red bin.

5. References

1. Application source code - https://github.com/praveendath92/token-bucket
2. Live demo of Token bucket - http://praveen.xyz/demos/token-bucket/
3. Token bucket Wiki - https://en.wikipedia.org/wiki/Token_bucket
4. Webstorm - https://www.jetbrains.com/webstorm/
5. Bootstrap - http://getbootstrap.com/
6. GreenSock - https://greensock.com/gsap
7. TweenLite - https://greensock.com/tweenlite
8. TimelineLite - https://greensock.com/timelinelite