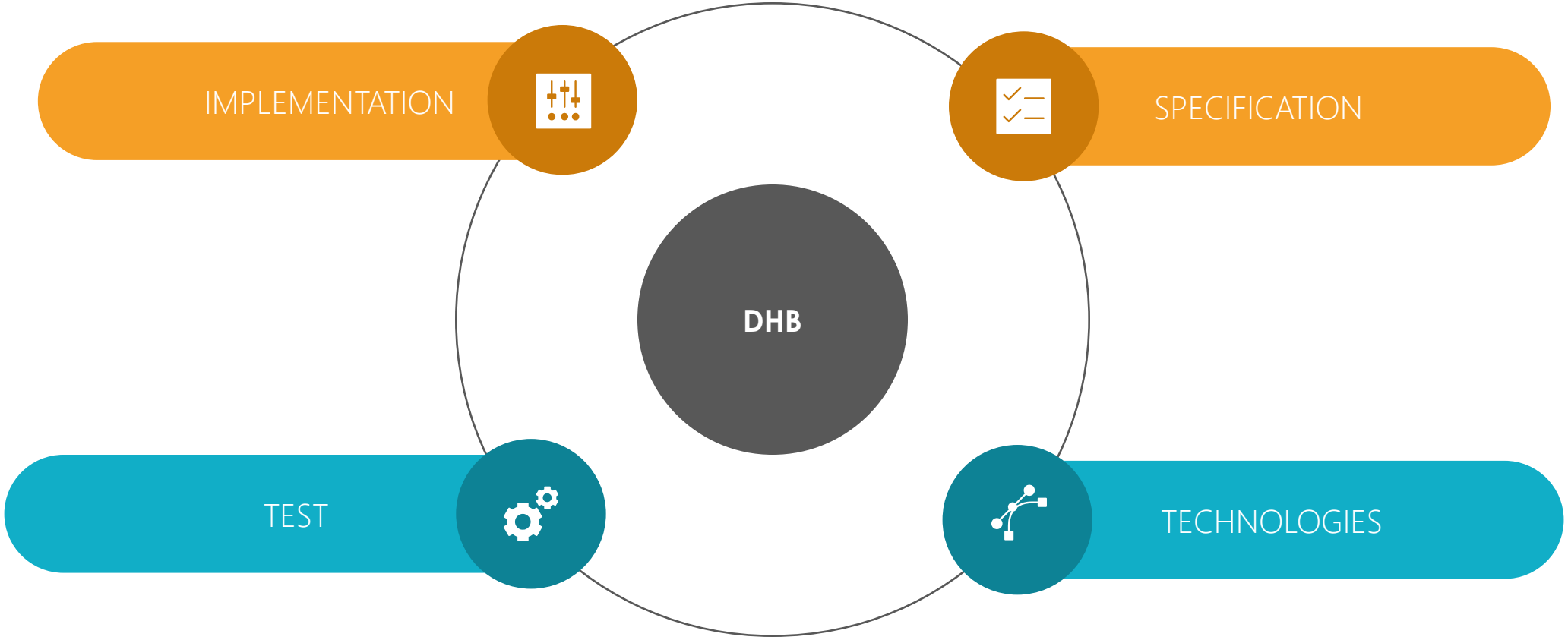
Distributed Hash-Breaker

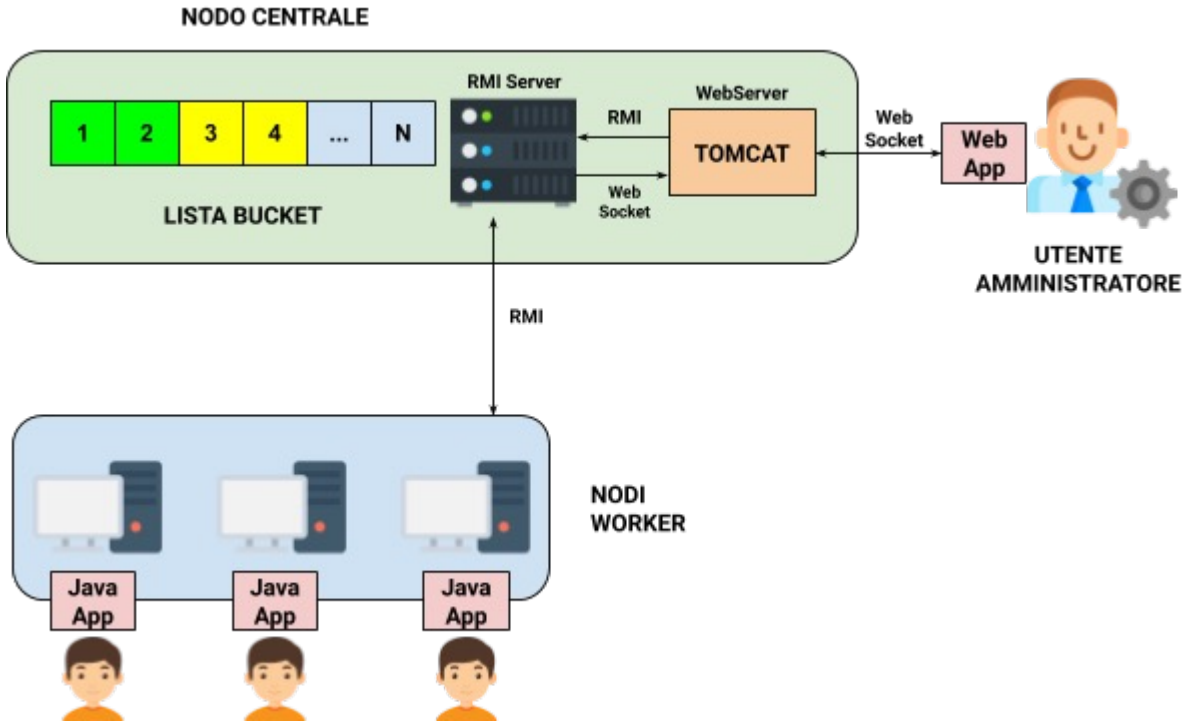
Concurrent and distributed Systems AY 2019/20

Filippo Scotto
Luigi Treccozzi

Outline



Specifications



- DHB is a distributed application that implements an attack on a Hash function, namely SHA-2
- It provides an **interface** through which users can join the attack
- The administrator can **plan the attack** and see the **attack progress**

Specifications

- **Administrator:** through the WebApp he can launch the attack, provide the plaintext to break, monitor the status of the attack
- **Workers:** Users that want to join the attack. Through GUI they are requested to provide a username

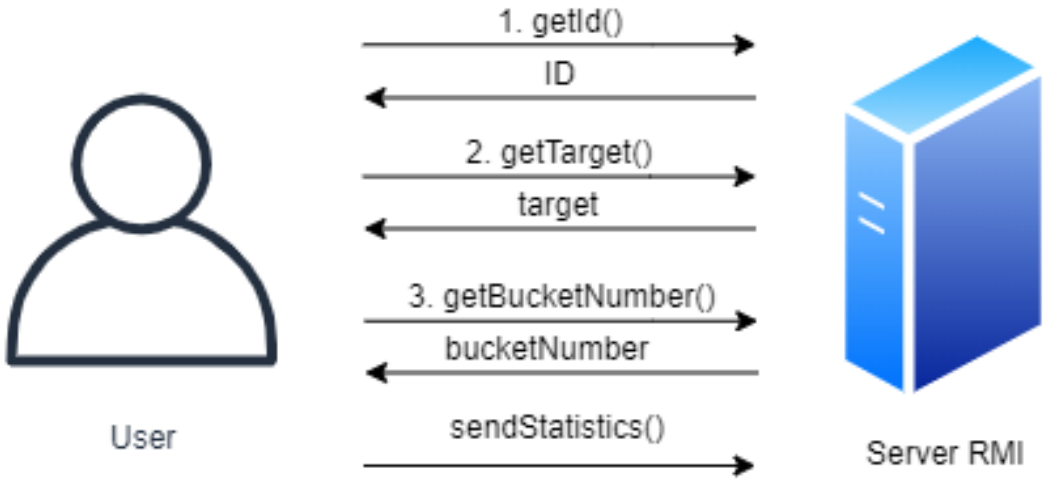
Used Technologies

- JDK-11.0.2 (Swing, RMI)

- MAVEN 3.6.1

- APACHE TOMCAT 9.0.27

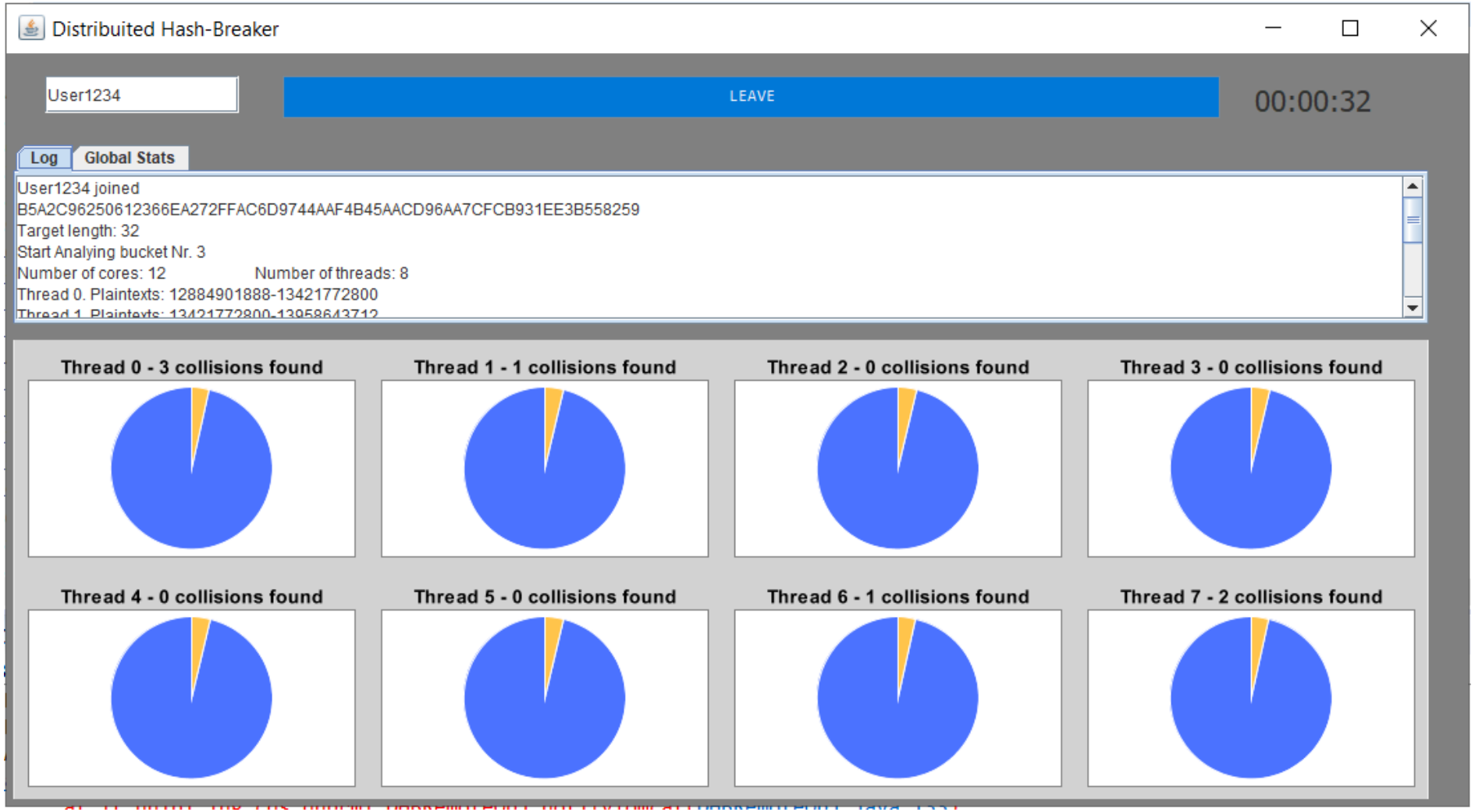
Client-Server Interaction



- Java Swing
- RMI
- Multi-thread
- xchart-3.6.0 [1]

[1] https://github.com/knowm/XChart

Client-Server Interaction



Client-Server Interaction

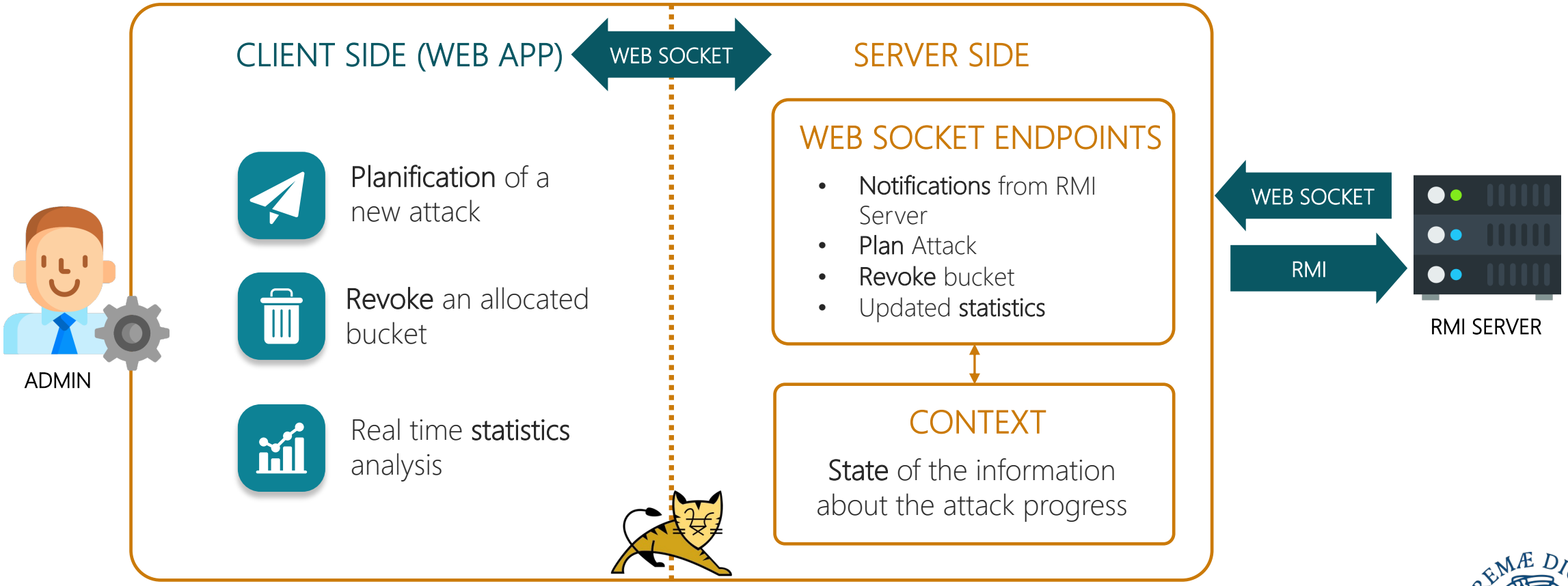
- When a user joins the attack a bucke is assagned
- This bucket is analayzed by various threads, in a number proportional to the number of cores available
- Periodically, these threads gather information about their work, and notify these updates to the server
- A user can monitor its own state of the attack: number of assigned bucket, percentage of completion, and the collisions that have been found

Server side operations

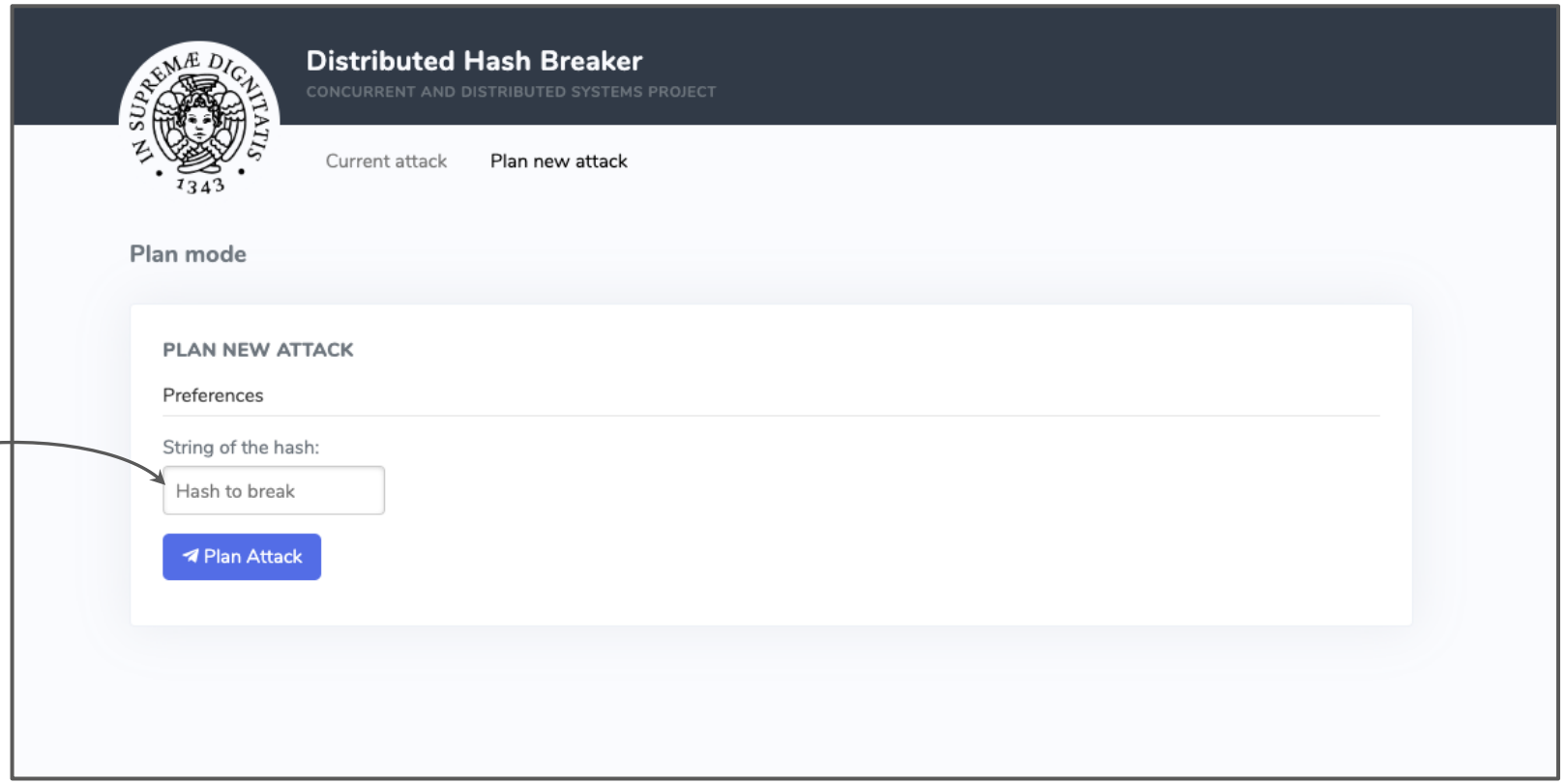
- Server handles a mutual exclusion bucket assignement
- Gathers information about the clients that joined the attack and update the statistics accordingly (Statistics thread)
- Makes sure that clients no longer active are canceled from the attack and their bucket are revoked and reallocated as free (Guardian thread)

Dashboard and WebServer

Here is presented the **overall architecture** used to implement the dashboard:



WebApp: Plan an Attack

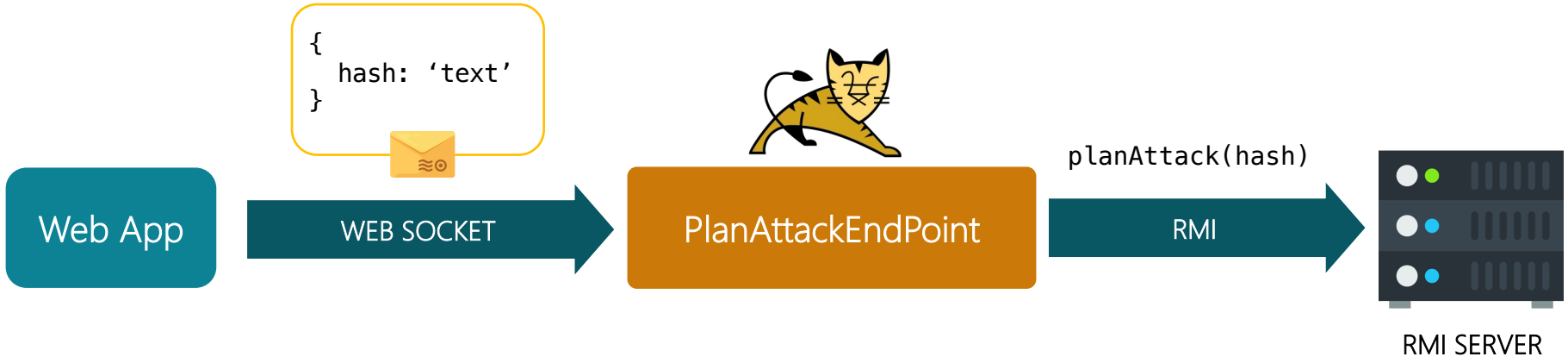


The plaintext to break

Once the system is ready, the first thing to do is **planning** the attack

WebApp: Plan an Attack

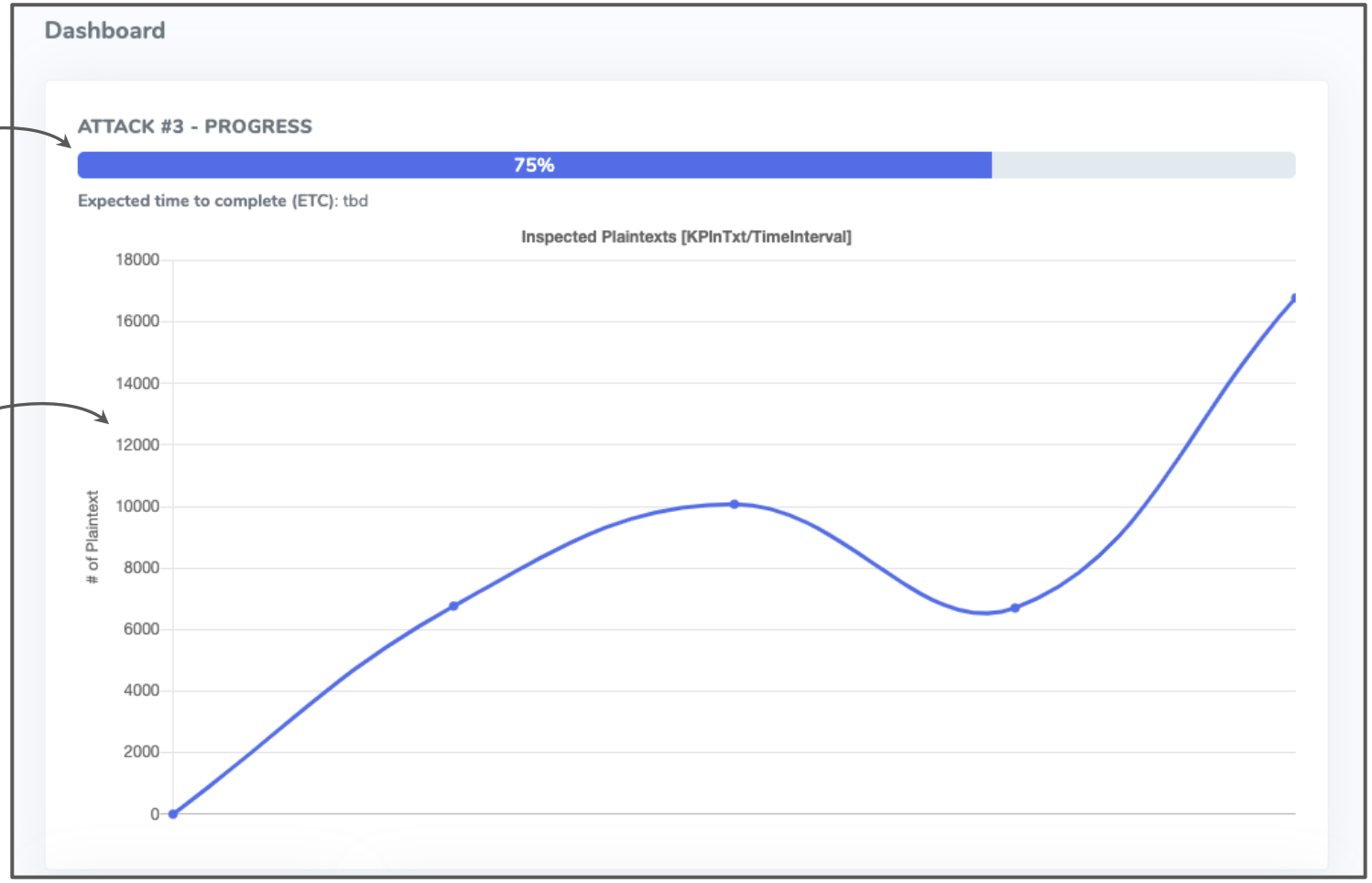
Once the administrator has decided which plaintext should be attacked, the WebApp sends a request to the **PlanAttackEndpoint** using a simple **JSON Message**. The WebServer will then ask the RMI Server to prepare a new attack.



WebApp: Global Statistics

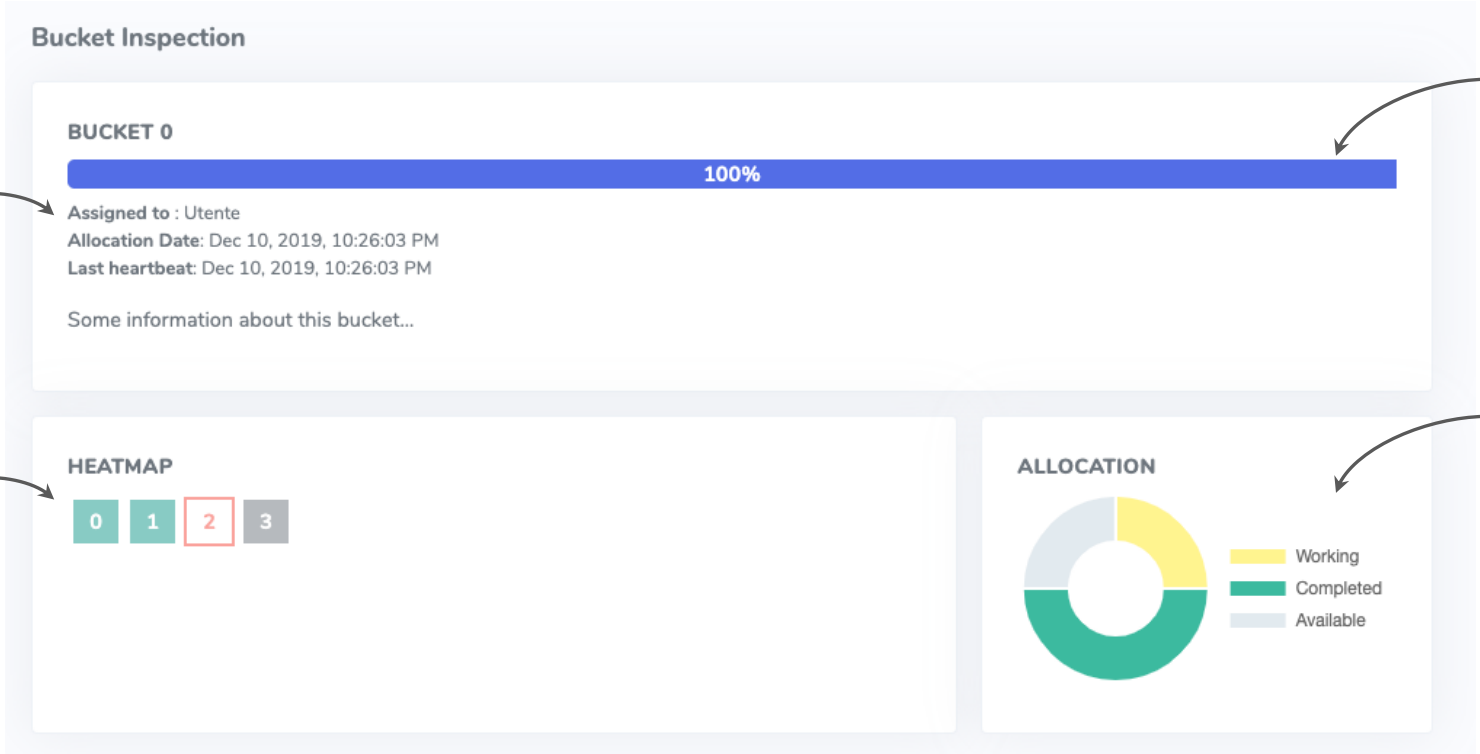
Overall progress

Graph of the number of Inspected plaintext



Operatively, the administrator can see the global **statistics** of the attack

WebApp: Bucket Statistics



Progress for bucket

Info about the user and last heartbeat

Bucket allocation chart

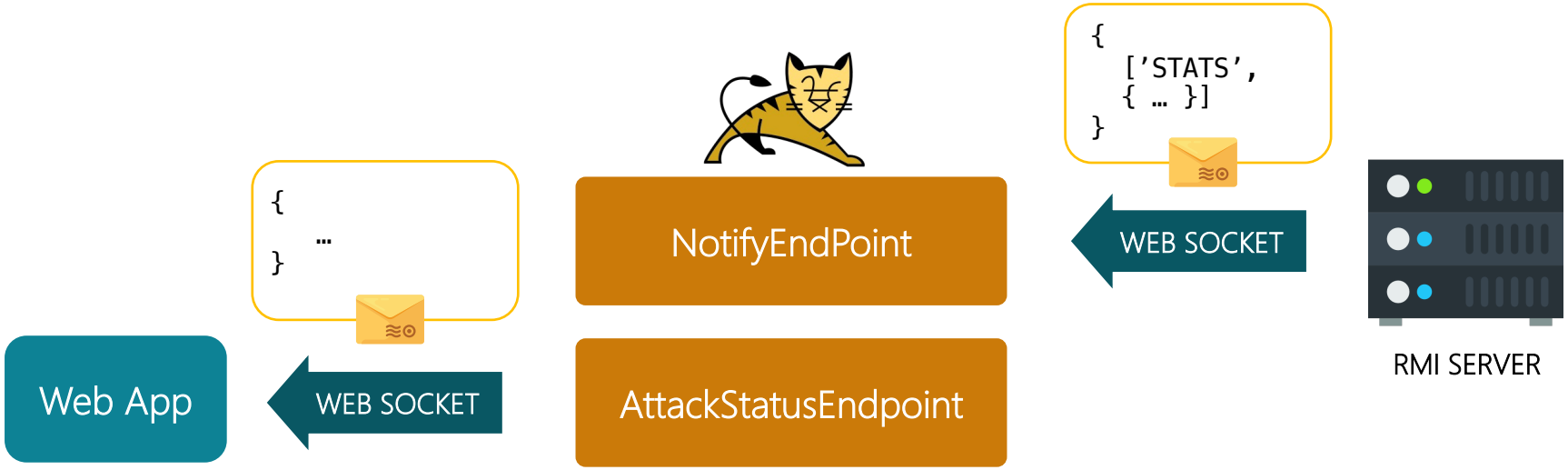
Heatmap of the buckets:

- Red: low percentage
- Yellow: Medium percentage
- Green: High percentage
- Gray: Not allocated

The administrator can also see the **statistics** for each allocated bucket

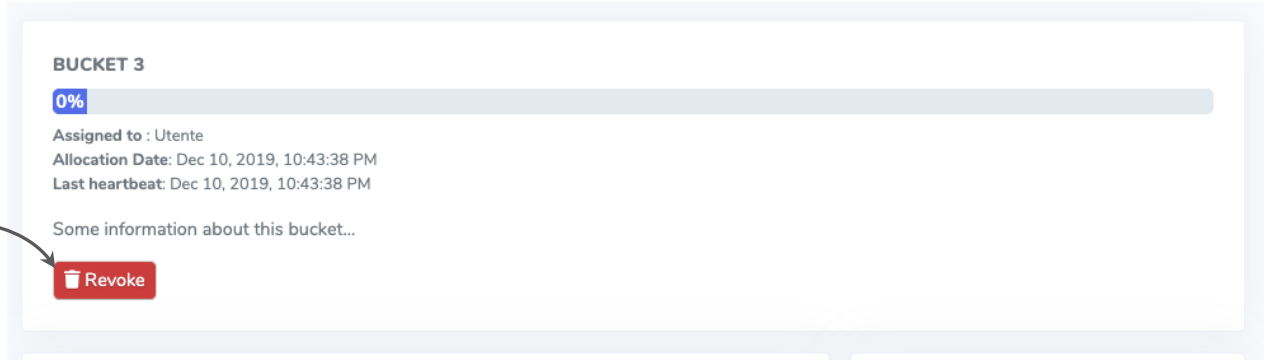
WebApp: Statistics

All the statistics are **periodically pushed** from the RMI Server (via a dedicated thread) using the **NotifyEndpoint**. The state in the **context** of the WebServer will be modified and the updates will be eventually propagated to the WebApp using the **AttackStatusEndpoint**.



WebApp: Revoke a Bucket

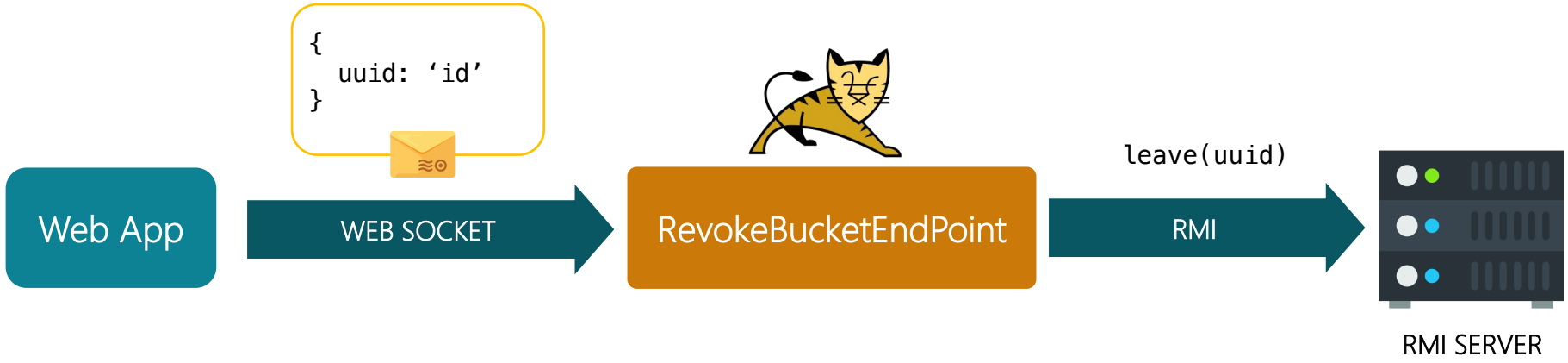
Revoke Button



The administrator can **revoke** a bucket that was allocated to a certain user

WebApp: Revoke a Bucket

Revoking a bucket is not much different from the previous operations:



Thanks