With the support of the Erasmus+ Programme of the European Union

EMJMD GENIAL

GREEN NETWORKING & CLOUD COMPUTING

UNIVERSITÉ DE LORRAINE

LEEDS BECKETT UNIVERSITY

LULEÅ UNIVERSITY OF TECHNOLOGY

Network programming and distributed applications (D7001D)

By

Md Abu Ahammed Babu

&

Mohammad Messbah Uddin

Luleå University of Technology

Department of Computer Science, Electrical and Space Engineering

Lab 4

Lab Objective:

- Create a multi-agent system, which in a coordinated manner performs actions to achieve certain goal.
- Create a scalable server architecture, which dynamically adapts to an increasing load and study its properties.

The Overall Scenario:

In this scenario, we increase the traffic load towards a TCP server in a controlled manner. The server is supposed to cope with the situation and perform dynamically. With this goal in mind, we developed a multi-agent system consisting of a Coordinator Agent (Architect) having the power to control the population and parameters of attacking agents through a GUI. The attacking agents periodically perform a certain action, opening a TCP socket towards a TCP server.

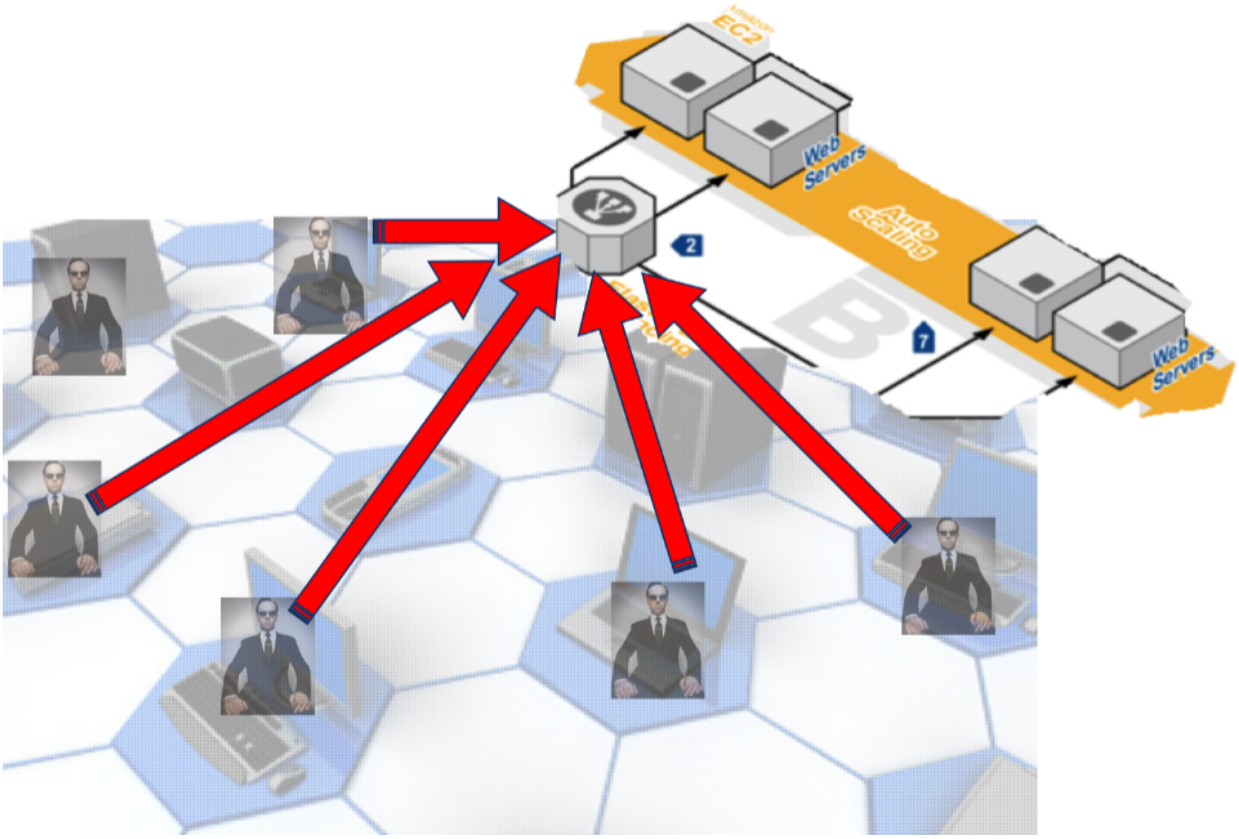
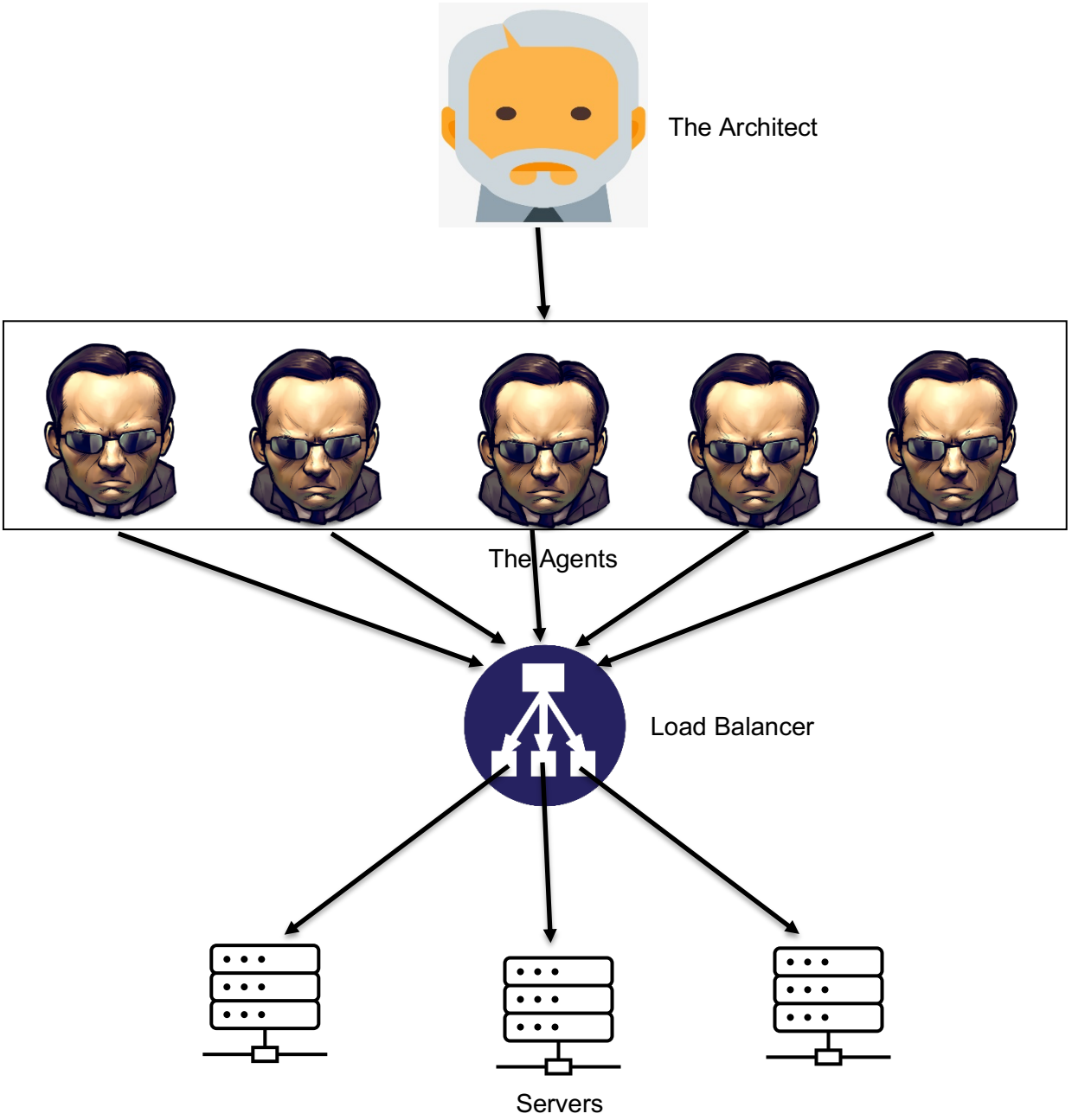


Figure 1: The Lab Scenario

System Architecture:



We have used three AWS account for Coordinator, Agent-Broker and the server respectively. Coordinator will command the broker agents to create more agents through a GUI interface. The Broker machine will get the parameters for the server IP to attack, port number, ticker duration and the number of agents need to create from this Coordinator GUI. After this it will create agents and initiate the attack. To make this architecture scalable we have used auto scale group in AWS account for Broker-Agent.

When the Coordinator gives command the broker agents will start to create agents and the auto scaling group has a threshold for CPU utilization as if its more than 70% of its capacity it will automatically add one more instance to create and launch the attack. However, when the rate of attack will be minimal that is if the CPU utilization goes down to 5%, it will remove 2 servers. This feature makes this application more scalable and reliant as it will provide single node failure tolerance and has the environment setup to launch more attacks through multiple servers.

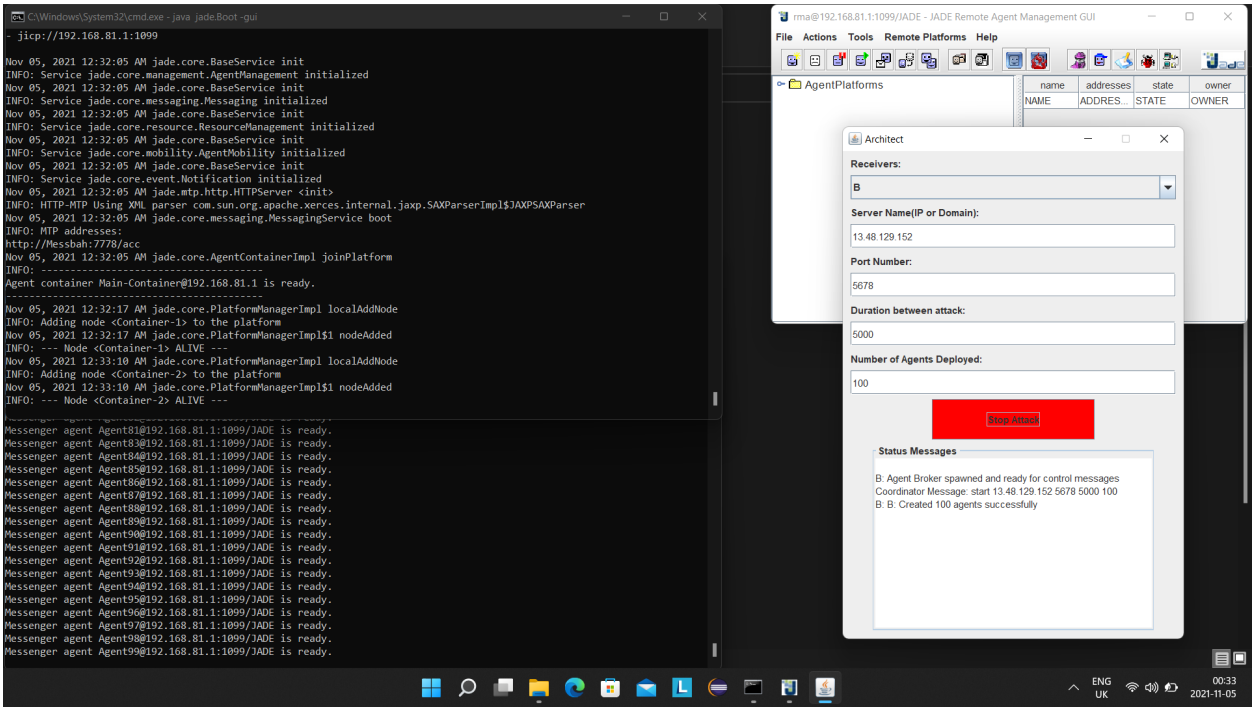


Figure 2: Agent Creation GUI

In the above figure, we created 100 agents to get Fibonacci sequence from the server. The next figure shows the server response.

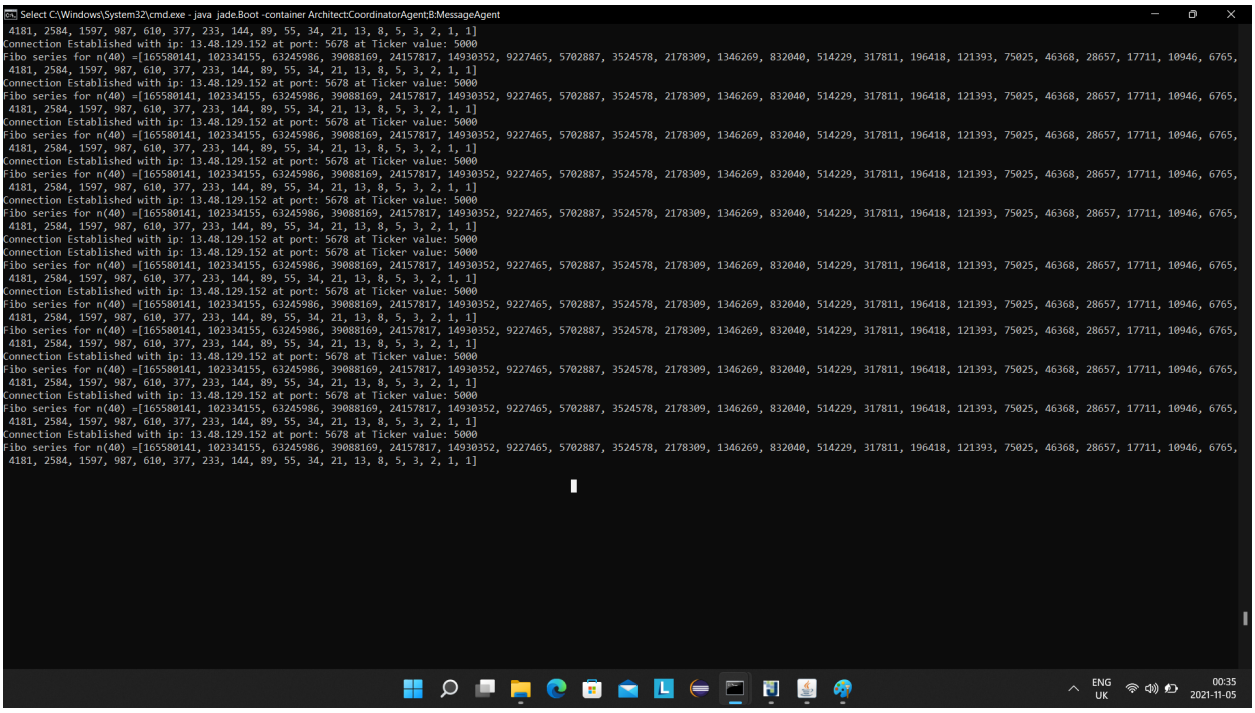
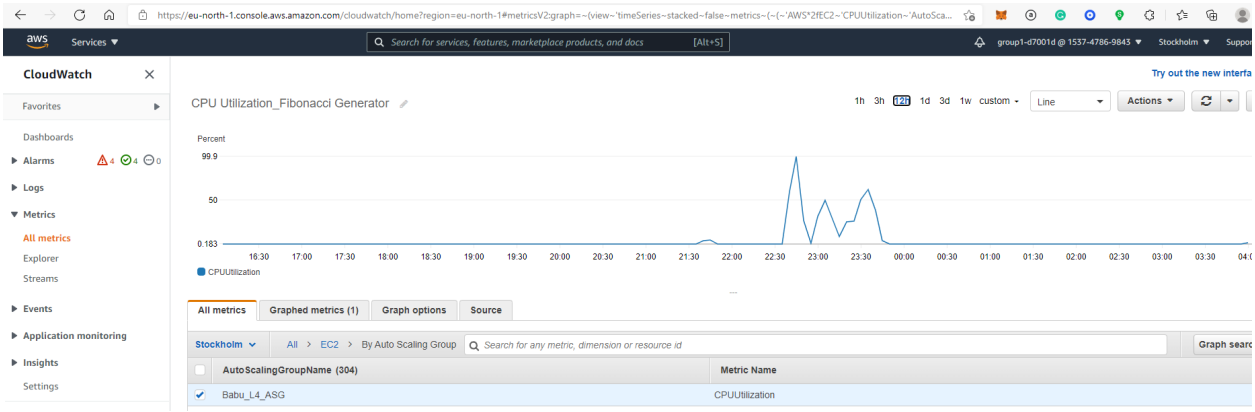


Figure 3: Reply from Server

Using CloudWatch We can see the CPU usage while the attack is made.



There is a load balancer which will redirect the http request that's coming from different agents or user to the server. It will save the server from getting too much task to handle at single instance. All the attacks from agent broker will come through this single point link (Load balancer DNS) and then distributed to all server nodes. As a result, the server will never go down and as it will proportionally distribute the tasks among several servers. There will be an auto scaling group for the server part which will deal with scalability problems. There is threshold to initiate more servers automatically if needed and we can also see the trends of CPU utilization using CloudWatch here.

Observations:

- We increased the number of agents gradually to see the behaviour how the server responds and the performance of our system. However, in our case, our system was unable to deal with 10000 agents.

- For small number of agents, the response from the server is very fast. However, the performance degrades with the increasing number.
- We learned how Distributed system, Auto scaling group works and when to use such system.