ISSN 1870-4069

A Web-based Didactic Tool for Teaching of Distributed Consensus

Abdiel González-Ortega, Francisco de Asís López-Fuentes

Universidad Autónoma Metropolitana-Cuajimalpa, Mexico City, Mexico
abditiger123@gmail.com, flopez@correo.cua.uam.mx

Abstract. Currently, educational models face the challenge of using information and communication technologies (ICT) to provide their students with the necessary tools and required knowledge in the 21st Century. The study of distributed systems is increasingly its important due to the popularity of web applications. For this reason, we address in this paper a topic known as the Byzantine Generals problem, which is a topic that has boomed in recent years, with the emergence of Bitcoin and Blockchain. This paper presents a web tool to support the teaching-learning process, which is given between the teacher and the students. The objective of this tool is to familiarize students with notions of consensus algorithms. Using our didactic tool users can visualize the operation of the Byzantine Generals algorithm and improve their knowledge about this consensus case.

Keywords: distributed systems, Web 2.0, byzantine generals, fault tolerance.

1 Introduction

Computer systems are already immersed in virtually all human activities. In particular, real-time systems are present in increasingly complex tasks and where an error can lead to catastrophic situations. Therefore, the fault tolerance capabilities of this type of systems are important for its success throughout its life cycle. Different characteristics of real-time systems must be converted to make them tolerant to failures, which is why there is a large field of research and development.

At present, the use of large distributed systems is becoming more common. For example, online services on the internet, cloud computing, internet of things and artificial intelligence are booming. Many of these systems are distributed systems that need to be scalable, because they constantly add and remove nodes (servers and clients). Distributed systems require being tolerant to failures to guarantee the service both to the current nodes and to those new nodes that want to join the system [1]. In this context, the concept of "distributed consensus" is very important to guarantee reliability in the system.

This paper presents a didactic tool to support the study of distributed consensus in the distributed systems. Our didactic tool simulates the case of Byzantine generals for different numbers of nodes with different numbers of traitor generals in order to know if the system can reach a consensus (agreement) [8].

The rest of this paper has the following organization. Section 2 gives an overview about related work, while an introduction to Byzantine general algorithm is given in Section 3. Design and implementation is given in Section 4. Operation and evaluation of our didactic tool is presented in Section 5. Paper concludes in Section 6.

2 Related Work

The main topics addressed in this work are two. First issue is about importance of information technologies for teaching-learning process and second issue is about the Byzantine Generals Problem. Several interactive tools can be found in the literature. Interactive Tools for Learning Sensor Network Basics [3] is a tool used to show the nodes in a network as intelligent autonomous sensors, which can measure certain characteristics of their environment, such as temperature, pressure, humidity, acceleration, etc. Wireless sensor networks [4] have a wide range of applications: climate monitoring, flood prevention, seismic monitoring, early detection of forest fires, etc. This tool is useful for master's students interested in wireless sensor networks as a research topic for their thesis, but who do not know enough to make a rational choice. This led the authors to start implementing an easy-to-use multimedia e-learning course, explaining the essential aspects of sensor networks and their routing protocols.

It has been observed that students, who have the opportunity to freely experiment with the systems learn faster and with more depth than those who do not have such opportunities. For that reason, a very simple interactive simulator that shows the principles of routing is included in the e-learning course.

One of the systems in which the fault tolerance of the Byzantine generals is being used is the Boeing 777 aircraft (through its ARINC 659 SAFEbus network) [5]. The flight control system of the Boeing 777 and the Boeing 787 flight control systems, use tolerance to Byzantine faults. Because these are real-time systems, Byzantine fault tolerance solutions must have very low latency. For example, SAFEbus can achieve tolerance to Byzantine faults with an order of one microsecond of added latency. The Boeing 777 is the first com-mercial aircraft manufactured by Boeing that employs the fault tolerance of Byzantine generals as the primary flight control system.

3 Byzantine Generals Algorithm

The problem [6] is that some of the generals are traitors and have as their objective that the armies be defeated. Therefore, our objective is to define an algorithm that allows loyal generals reach a consensus on the action plan. The final decision will be by majority vote on the initial choices. If they tie, the decision is withdrawn. The problem assumes that the generals are computers and the messengers are communication channels. Generals may fail, but not messengers. The requirements to obtain consensus are:

- Termination: all non-defective processes must decide [2].
- Agreement: the final decision of each non-defective process must be identical.

– Validity: if every non-defective process starts with the same value (V), its final decision must be (V).

The problem of the Byzantine generals is frequently referred in the failure tolerance, and it is also known as a Byzantine problem. Lamport et al. [2,7] demonstrated that in a system with *k* defective processes, consensus can be achieved only if *2k + 1* processes are functioning correctly, for a total of *3k + 1* processes.

3.1 Case with Three Generals

Figure 1 shows the basic example of Byzantine Generals problem. This scenario consists of three generals, which are represented in a hierarchical way. There is a commander (there can only be one) who sends the orders to the general lieutenants that in this first case we have two lieutenants. We note that the generals with the rank of lieutenant cannot send messages to the commander or return those received, because the final agreement is carried only with the generals of the same rank as lieutenants and only if they are loyal generals. In this scenario, three nodes are shown as loyal generals. We see that the general who has the commander hierarchy is who sends the message to the other lieutenants in this case Lieutenant1 and Lieutenant2. The messages that the commander sends are "Attack" given that the commander is a loyal general.

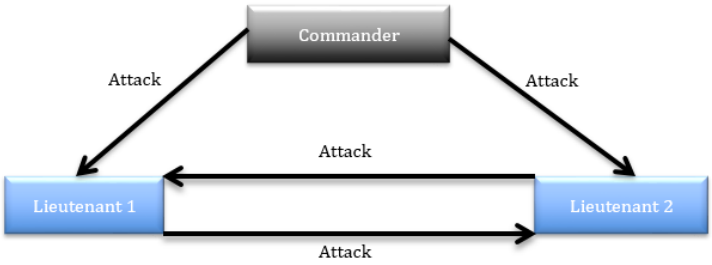

Fig. 1. Basic model of the Byzantine generals.

Consequently, Lieutenant generals send messages to each other because, as previously explained, the Lieutenant generals cannot communicate with a superior general. Lieutenant1 sends Lieutenant2 the message received by the Commander and consequently Lieutenant2 sends Lieutenant1 the message he received. In this scenario it is assumed that all generals are loyal.

As shown in Figure 1 we have the base case which consists in that all generals are loyal. The general who will be responsible for deciding if consensus is reached will be one of the loyal lieutenants, never a traitor general will have that responsibility to make the decision final in this case can be both Lieutenant1 and Lieutenant2 since both are loyal. After receiving the message from the commander and the lieutenants have finished sending their messages, an evaluation is made of how many are in favor of attacking and how many in favor of retreat and depending on the results of those messages will be how an agreement is reached. In this case, we have that the Lieutenant1 will receive the message from the commander who has sent a message to

"Attack" and the other from the Lieutenant2 who in turn also sends the "Attack" message, resulting in the Lieutenant1 reaching an agreement to attack.

Now, we analyze what happen when we have a traitor general in the group. Figure 2 shows this scenario. Communication between generals is same and the hierarchy of them is also maintained, the only variant in the scheme is the traitor node. The Commander sends the message "Attack" to the two Lieutenants, but here we see that one of them is a traitor in this case the Lieutenant2. Both lieutenants have already received the message. Then, both nodes exchange its messages together. Lieutenant1 as a loyal general sends the message "Attack" received from the Commander. However, Lieutenant2 is a traitor, then this general changes the message received from the Commander and sends a false message ("Retreat") to Lieutenant1. In this scenario, each Lieutenant receive two messages, one from Commander and the other from another Lieutenant. We can see that Lieutenant 1 receives from Commander the order to "Attack", while from the traitor Lieutenant receives the order "Retreat", resulting in a tie. In this case is not possible to reach an agreement between the nodes. Then, we will analyze a case with four nodes.

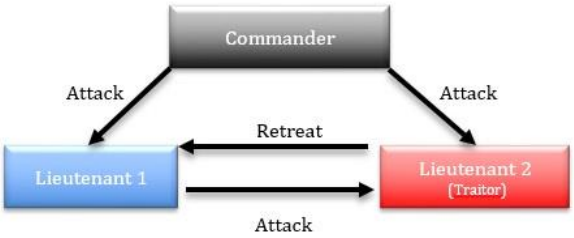

Fig. 2. Case three generals with a traitor (consensus is not possible).

3.2 Case with Four Generals

In this scenario, we have four nodes, the hierarchy always remains. Now, there are three Lieutenant generals and a Commander. Commander sends the message (attack) correctly to each of the lieutenants, and after this each lieutenant communicates this message to each other. Thus, Lieutenant1 sends to Lieutenant2 and Lieutenant3 the message "attack" received from the commander. Lieutenant2 and Lieutenant3 performs similar operation. In this case, there is no problem in reaching a consensus. Now, we analyze what happen when we have a traitor general in the group. The analysis is carried out in parts in order to understand well if consensus is reached or not. In the first part, the Commander sends the "Attack" message correctly to the three Lieutenants, as we can see in Figure 3. However, Lieutenant3 is traitor. Thus, Lieutenant1 and Lieutenant2 forward the message to the other two lieutenants as it was received from the commander, but Lieutenant3 changes the "Attack" message received form the Commander and sends a "Retreat" message to Lieutenant1 and Lieutenant2. Each Lieutenant must take a decision based on the number of messages received. For example, Lieutenant1 receives three messages, one from the Commander and two from the Lieutenants, but one of them is incorrect. Lieutenant1 has two "attack" messages and a "Retreat" message (from traitor general), so as the majority of messages are "Attack", then the consensus is carried out to attack. A similar way is for Lieutenant2. We can see that Commander, Lieutenant1 and Lientenant2 can reach a consensus.

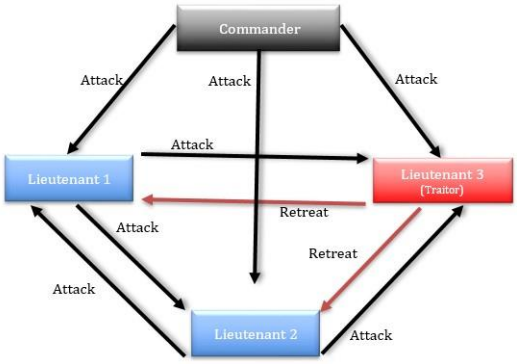

Fig. 3. Communication of the four generals and evaluation (consensus is possible).

4 Design and Implementation

This section gives a briefly description about design and implementation of our web tool. Figure 4 shows a sequence diagram which describes our system for different types of users: a student, a teacher and a general user. Users can see the options shown in the menu and they can choose one of those. In start section a brief description of the origin of this problem is shown. In problem section an explanation about algorithm operation is presented. Finally, in algorithm part we find the Byzantine general model. There each user can perform tests, and analyze the results.

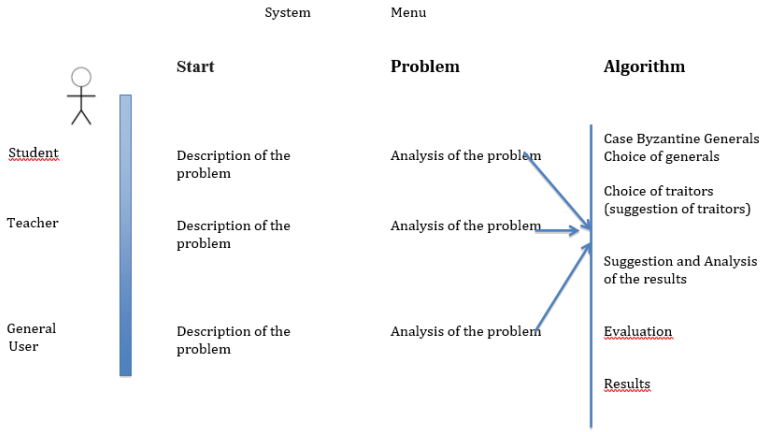

Fig. 4. Sequence diagram.

To implement our design, we are used several programming tools such as JavaScript (Vis.js library), HTML and CSS. Our implementation considers the following tasks

– **Draw function.** This function creates the array that will contain the nodes, and these are assigned values to each of the variables, which will be as these are identified in the program.

- **Commander and Lieutenant Node**. After having created the node to which we are going to connect our other generals, we create our Lieutenants. We use a "FOR" cycle to perform the evaluation according to the number of Lieutenants requested by the user. Once nodes have been created, the connection is made with each one of them. We need to avoid overlapping or some inconvenience in the program.
- **Restrictions.** In order to program works according with the base formula, it was necessary to make the restriction indicated from design, which is the maximum number of traitors that can be in each of the scenarios.
- **Check function** helps to the user to know the maximum of traitor generals that can be introduced to reach an agreement to attack. This message informs to a user the maximum number of traitor generals to have a consensus of "Attack", but if the user decides to enter another amount the program does not restrict doing this.

5 Operation and Evaluation

To evaluate our web tool, we organized its operation in three parts:

1. **Start.** In this section a detailed description of the algorithm is made as well as the analysis of the equation. The base cases that were analyzed in the project are proposed too. Figure 5 shows this scenario.

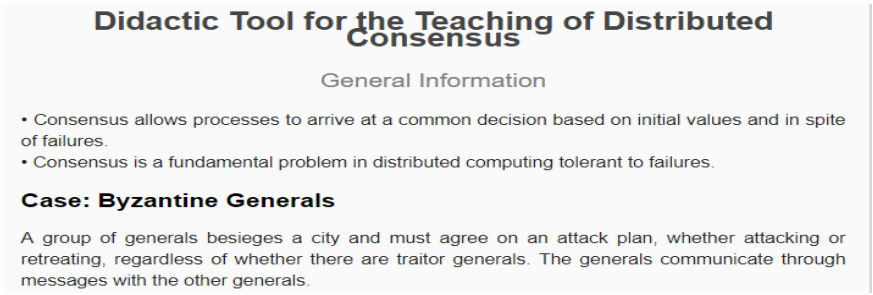

Fig. 5. A part of home section of our web page.

2. **Issue.** Figure 6 shows the part where the user analyzes the algorithm of the problem, similar to the analysis made in Section 3.
3. **Algorithm** is the graphic and interactive part that the user can see. Figure 7 gives an example about the program that indicates the maximum amount of traitor generals that the user can enter. It should be noted that this does not limit the user to enter more, if not, which is a suggestion for the user to obtain a consensus to attack.

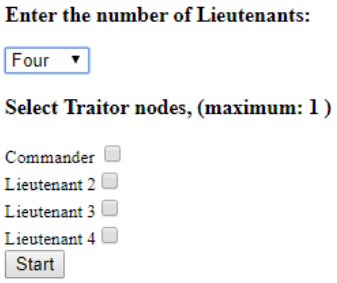

Fig. 6. Description of the problem to be retreated.

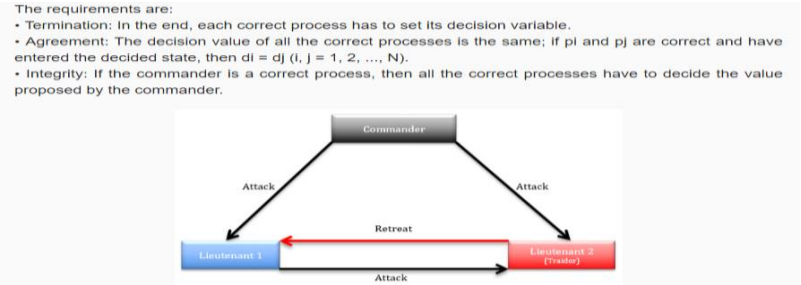

Fig. 7. Selection of traitorous nodes and suggestion.

The user has the option to choose the number of generals to be shown on the platform. A user can also choose how many traitor nodes must appear and at the end of the evaluation a table with the final result is shown. We can see in Figure 8 that result is "CONSENSUS ATTACK". When many nodes are evaluated a graphical representation becomes complex, and our tool uses a table with status of all participant nodes to deal with this problem.

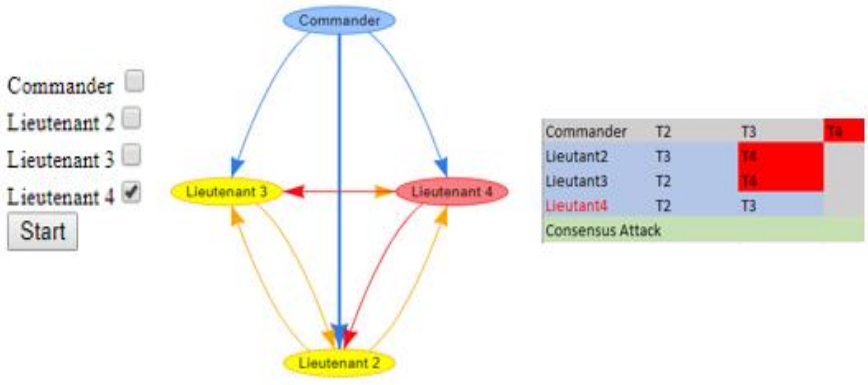

Fig. 8. An example for four generals using our didactic tool.

6 Conclusions

This paper presents a didactic tool to help understand the distributed consensus, in particular, the Byzantine Generals problem. In addition, we tested the use ICT as facilitator in the teaching process for the new generations of students who are now more familiar with new devices to access information in Internet. We hope that our didactic tool also helps to disperse the knowledge of a simpler and cheaper way to those places where is difficult to acquire a large number of books. Our didactic tool has had a good acceptance in our academic community because it solves the Byzantine Generals problem of a graphical an interactive way. We think that with the growth of the Internet this type of didactic tools will be very important in the teaching/learning process. As future work, we plan to add a database to our didactic tool in order to storage information about user experience (UX). In such a way that using artificial intelligence techniques we could increase emphasis on knowledge representation as an activity within a perceptual space and organized by social interactions.

References

1. Kindberg, T., Dollimore, J., Coulouris, G., Blair, G.: Distributed Systems. Pearson Education Limited, pp. 5–25 (2013)
2. Lamport, Shostak, R., Pease, M.: The Byzantine Generals Problem. ACM Transaction on Programming Languajes and Systems 4(3), pp. 382-401 (1982)
3. Uwase, M.P., Tiberghien, J., Steenhaut, K.: Interactive Tools for Learning Sensor Network Basics. In: Advances in Engineering and technology. MacMillan UGANDA, pp. 148–154 (2011)
4. Wein, J., Kourtchikov, K., Cheng, Y., Gutierrez, R., Khmelichek, R., Topol, M.: Virtualized Games for Teaching about Distributed Systems. In: Proceeding of the 40th ACM Technical Symposium on Computer Science Education, Chattanooga, TN, USA, pp. 246–250 (2009)
5. Yeh, Y.C.: Design Considerations in Boeing 777 Fly by Wire computers. Boeing Commercial Airplane. In: Proceedings Third IEEE International High-Assurance Systems Engineering Symposium, Washington, DC, USA, pp. 64–72 (1998)
6. Arévalo-Viñuales, S.: Tolerancia a fallas en sistemas distribuidos mediante replicación de procesos. Tesis Doctoral, Universidad Politécnica de Madrid (1988)
7. Pérez-Porto, J., Merino, M.: Definición de consenso. http://definicion.de/consenso, Last accessed 2018/08/20
8. Jiménez-Peris, R., Patiño, M., Jiménez-Merino, E.: Consenso. Obtained from: http://lsd.ls.fi.upm.es/lsd.html. Last accessed 2018/08/20.