The Eurasia Proceedings of
Educational & Social Sciences (EPESS)
ISSN: 2587-1730

**The Eurasia Proceedings of Educational & Social Sciences (EPESS), 2016**

**Volume 5, Pages 42-46**

**ICRES 2016: International Conference on Research in Education and Science**

# IMPLEMENTING THE DISTRIBUTED BREADTH FIRST SEARCH ALGORITHM IN OMNET++ FOR TEACHING AND LEARNING PURPOSES

Esranur Galip
Ege University, Computer Engineering Department

Hasan Bulut
Ege University, Computer Engineering Department

**Abstract**: A distributed system is considered as a set of computers communicating through the network and running collaboratively to coordinate their activities and to share the resources of the system to achieve a common goal. The coordination is achieved by exchanging messages, which carry information. Distributed algorithms play a crucial role in this coordination. However, teaching and learning distributed algorithms is difficult due to the inherent complexities of the distributed system. Since it is costly to construct a network of computers to run distributed algorithms to conduct research, teach and learn, many commercial and freely available open source simulation tools have been developed for simulating network systems and hence, distributed systems. These tools facilitate the development of distributed algorithms for different environments. One of these tools is OMNET++, which is a component-based C++ simulation library and framework for building network simulators and offers a graphical runtime environment.To facilitate the understanding of the working mechanism, a distributed system can be modeled as a graph. Each computer in the distributed system is represented by a vertex, called node and a link between two computers is represented by an edge. Hereby, many graph algorithms can be utilized within a distributed system. For instance, traversal of computers (nodes) in a distributed system is important and used for solving many problems. Many algorithms provide traversal of nodes. In this study, we would like to demonstrate the use of a simulation tool for teaching and learning one of the fundamental distributed graph algorithms called Breadth First Search (BFS) algorithm. We use OMNET++ to visualize the steps of constructing a BFS tree, where colors of edges are dynamically changed to indicate the inner workings of the algorithm. In addition, a learner can visually trace the flow of the messages between nodes in the simulation.

***Keywords*:** Distributed algorithms, graph algorithms, breadth first search algorithm, simulation for education, omnet++

## Introduction

In computer science (CS), everything is a part of dynamic process and consists of abstract concepts whose imagination is so complex and hard. The process of an algorithm's behavior is dynamic, which cannot be visualized using static techniques such as images or flow charts. Let's consider a topic, which requires to be imagined step by step. Most of lecturers explain the topic verbally without any visual material. Thereby, many students spend hard times to solve the meaning of the long and complex sentences, which emphasize the core of the course. In that case, visualization plays important role for understanding the topic clearly. However, visualization can significantly affect CS education (Thomas Naps, 2003). For this purpose, Algorithm Visualization (AV) has been used since 1981 in CS education. Many educational AVs are being implemented and freely distributed for educational purposes. Jeliot (University of Eastern Finland, 2016) is one of them. It was developed to help high school students to understand and learn Java programming. It visualizes the source code by generating animation, which shows behavior of the program (Fouh, Akbar, & Shaffer, 2012).



- Selection and peer-review under responsibility of the Organizing Committee of the conference

*Corresponding author: Esranur Galip- E-Mail: esranur.galip@ege.edu.tr

Distributed system is a dispensable part of CS. A distributed system is considered as a set of computers communicating through the network and running collaboratively to coordinate their activities and to share the resources of the system to achieve a common goal. The coordination is achieved by exchanging messages, which carry information. In the coordination of processes in the system, distributed algorithms play a crucial role. Designing a distributed algorithm is more complex than designing a sequential algorithm because of considering many concurrent events in the distributed case. Hence, teaching and learning distributed algorithms is difficult due to the inherent complexities of the distributed system, which consists of multiple independent components, where each has a separate state and control mechanism. Since it is costly to construct a network of computers to run distributed algorithms to conduct research, teach and learn, many commercial and freely available open source simulation tools have been developed. These tools facilitate the development of distributed algorithms for different environments and are so important to be used for algorithms simulations. The research of (O'Donnell, 2006) indicates that the simulation of an algorithm provides learners to think with a higher level for the algorithm's behaivour. For this purpose, (Abdou, Abdallah, & Mosbah, 2014) developed Java based ViSiDiA. With developing ViSiDiA, they aim to help teaching distributed algorithms and contributing to the research activities. The tool supports implementing, simulating and visualizing distributed algorithms for several types of networks.

OMNET++ is a component-based C++ simulation library and framework for building network simulators and offers a graphical runtime environment (OMNET++, 2016). In addition, it is extensible and offers an Eclipse-based IDE that is commonly known among researchers and students. Since users do not need to spend extra time to learn the environment, it is easier to be accepted by users. Its graphical interface enables users to build a network by using network components easily. There are extensions for real-time simulation, network emulation, etc. Its visualization for communications and transmission of messages facilitates the understanding of abstract concepts. Hereby, it is currently gaining widespread popularity in the scientific community. In this study, we would like to demonstrate the use of a simulation tool for teaching and learning one of the fundamental distributed graph algorithms called Breadth First Search (BFS) algorithm. We use OMNET++ to visualize the steps of constructing a BFS tree, where colors of edges are dynamically changed to indicate the inner workings of the algorithm. In addition, a learner can visually trace the flow of the messages between nodes in the simulation.

This paper is in detail as follows. Section 2 covers the implemented algorithms and methods. Results are discussed in Section 3. Conclusion is covered in Section 4.

## Methods

Distributed system is an important area of CS. A distributed system can be modeled as a graph. Each computer in the distributed system is represented by a vertex, called node and a link between two computers is represented by an edge. Hereby, many graph algorithms can be utilized within a distributed system by converting them into distributed versions. The basic distributed graph algorithms can be stated as Breadth First Search (BFS), Depth First Search (DFS), Topological Sort, Shortest Path Algorithms, Minimum Spanning Tree Algorithms, etc. In this study, we examine one of the fundamental graph algorithms, distributed synchronous Breadth First Search (BFS) algorithm. The algorithm constructs a BFS tree for a source node. Hereby, each node's distance to the source node is determined (Raynal, 2013). The examined BFS algorithm is implemented and simulated in OMNET++ simulation tool.

### Synchronous Breadth First Search Algorithm

In synchronous BFS algorithm, the source node initiates BFS algorithm and undertakes the synchronizer task. At the end of the algorithm, BFS tree is constructed, where the distance of each node from the source node is found. The source node uses different message types for synchronizing and finding parent and child relations. The source node starts discovering nodes in each layer by sending a GO message with layer id. The GO message is interpreted by the receiver node in two aspects. If it is received for the first time, the node is discovered by the sender node. The sender node is the parent of the receiver node. From this moment, all messages from the source node will come through its parent. If the receiver node has neighbors, it informs its parent about the discovery of the next layer. We can state this node as an active node. Therefore, the edge between parent and child is colored as red. If the receiver node has no neighbor, it informs its parent about its termination of the algorithm and the edge between parent and the child is colored as green. We can state this node as a passive node. Hereby, the states of the nodes in the system can be observed easily. If the node is active and if it receives a GO message from its parent again, the node will discover nodes in the next layer or if it has already discovered all of its children, the node undertakes the intermediary role. Then, it will forward the message to its active children to inform them about the discovering nodes in the next layer. In each iteration, parents gather responses from their

children about the completion of the discovering nodes for the layer. Then each parent forwards this message to its parent until the message is received by the source node. After that, the source node initiates the discovery of the next layer by sending a GO message with next layer id. If a parent gathers messages for the termination of the algorithm from all its children, it forwards the message to its parent. Hereby, the subtree is discovered and edges between parents and children are colored as green in the subtree. When the source node gets messages for the termination of the algorithm from all of its children, the source node realizes that the desired BFS tree is constructed.

**Implementations**

In OMNET++, if the developer aims to build a network, first, she should define submodule(s), which will be a component of the network. For instance, in synchronous BFS algorithm implementation, to represent each node, *node* submodule is defined in the network. Using the *node* submodule with existing modules, such as a channel module, a connection is established between the components of the network. Design of the network can be expressed as a ned file, which can also be defined by using a graphical built-in interface. This facilitates the topology design easier for developers. A snapshot of the initial state of the network is given in Figure 1. The topology is generated with a built-in random topology generator, where the nodes are derived from the implemented *node* submodule.

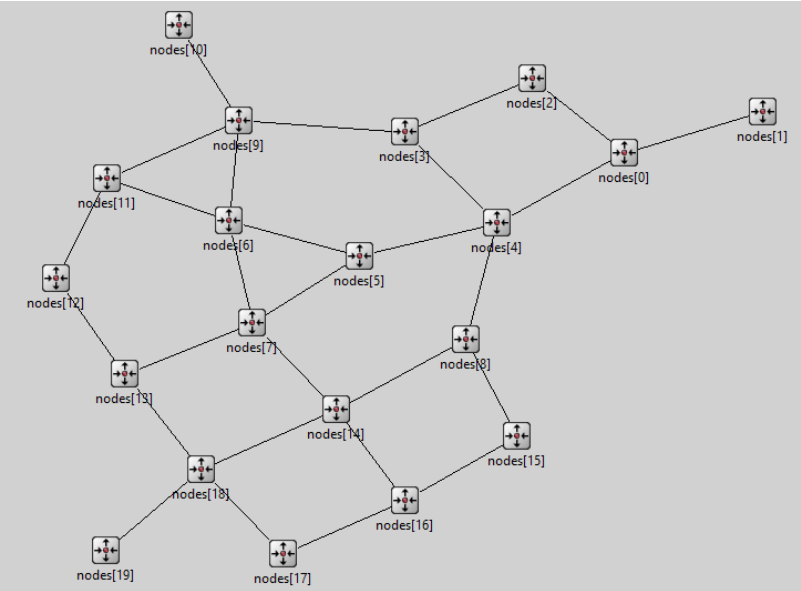


Figure 1. The snapshot of the initial state of the example network

## Results and Findings

In Figure 2, a snapshot of BFS algorithm's simulation is given. To make the algorithm's behavior clear and enable students to understand the working mechanism of the BFS algorithm, we have colored edges during the discovery of layers. If the child has just been discovered by its parent, the edge between the parent and the child is colored as red. If the child node finishes searching, it means the child may be a leaf or its neighbors have already been discovered by other nodes, the edge between child and parent is colored as green as shown in Figure 2. For instance, *nodes[10]* and *nodes[1]* are leaves and they are discovered by their parents, *nodes[9]* and *nodes[0]*, respectively. *node[6]* and *node[15]* could not discover their own neighbors as their children before their neighbors. Each message in the simulation is indicated with a message name and a circle filled in red. As seen in the snapshot of the simulation in Figure 2, *nodes[0]* sends a GO message to its children to inform them about the discovery of nodes in *layer5*.

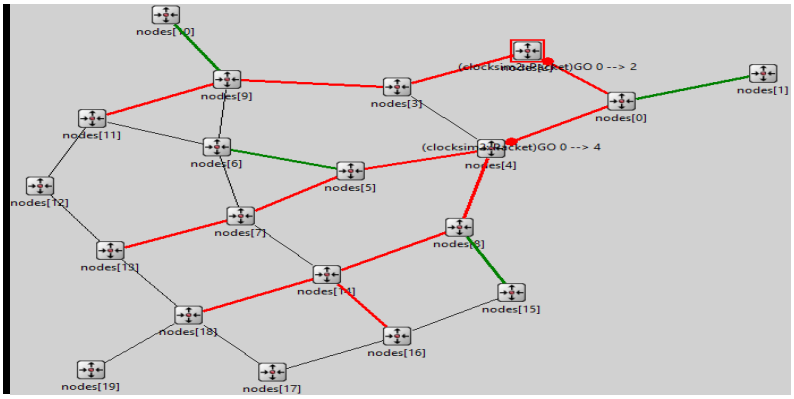


Figure 2. The snapshot of discovering process for bfs algorithms

The output of the BFS algorithm for the input topology shown in Figure 1 is a BFS tree, where edges between parents and children are colored as green as shown in Figure 3. By implementing the BFS algorithm in OMNET++, we visualize the inner workings of the BFS algorithm while reaching the final BFS tree.

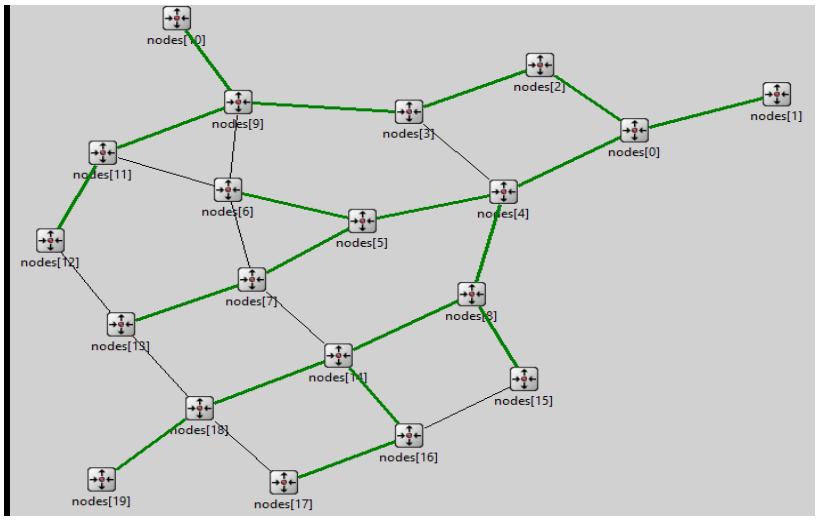


Figure 3. The final BFS tree

## Conclusion

Visualization is important in teaching and learning distributed algorithms because of the inherent complexities in a distributed system. Many students spend too much effort to solve the meaning of the long and complex sentences that explain the algorithms' behaviours. In addition, lecturers have difficulty in explaining the abstract concepts verbally. For this purpose, in this study, we aimed to demonstrate the use of a simulation tool for teaching and learning one of the fundamental distributed graph algorithms. As an example for distributed graph algorithm, we decided to implement the synchronous version of the distributed BFS algorithm, since it is easier to understand and implement. As a simulation tool, OMNET++ was preferred due to its support for visualization. We are able to visualize the steps of constructing a BFS tree, where colors of edges are dynamically changed to indicate the inner workings of the algorithm. In addition, a learner can visually trace the flow of the messages between nodes in the simulation.

## References

Abdou, W., Abdallah, N., & Mosbah, M. (2014). ViSiDiA: A Java Framework for Designing, Simulating and Visualizing Distributed Algorithms. *IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 43-46.

András Varga, R. H. (2008). An overview of the OMNeT++ simulation environment. *In Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 60. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering.

Easley, D., & Kleinberg, J. (2010). *Networks, Crowds, and Markets: Reasoning about a Highly Connected World.* Cambridge University Press.

Fouh, E., Akbar, M., & Shaffer, C. A. (2012). The Role of Visualization in Computer Science Education. *Computers in the Schools*, 95-117.

O'Donnell, F. (2006). Simulation Frameworks for the Teaching and Learning of Distributed Algorithms. *Thesis of Doctor of Philosophy*. University of Dublin, Trinity College.

OMNET++. (2016). OMNeT++ Discrete Event Simulator. Retrieved March 15, 2016 from https://omnetpp.org/

Raynal, M. (2013). *Distributed Algorithms for Message-Passing Systems.* London: Springer.

Thomas Naps, S. C. (2003). Evaluating the educational impact of visualization. *ACM SIGCSE Bulletin, 35(4)*, 124-136.

University of Eastern Finland. (2016). *Jeliot3*. Jeliot3. Retrieved March 15, 2016 from https://cs.joensuu.fi/jeliot/downloads/jeliot372.php

Varga, A. (1999). Using the OMNeT++ Discrete Event Simulation System in Education. *IEEE Transaction on Education, 42(4)*, 372.