DOI: https://doi.org/10.33103/uot.ijccce.23.3.10

A Survey of Graph Techniques Applied in Software Defined Networking

Sama Salam Samaan[1], Hassan Awheed Jeiad[2]
[1,2]Computer Engineering Department, University of Technology, Baghdad, Iraq
[1]sama.s.samaan@uotechnology.edu.iq, [2]hassan.a.jeiad@uotechnology.edu.iq

Abstract—*Modelling computer networks in general, particularly Software Defined Networking (SDN) as a graph, is beneficial in network planning and design, configuration management, traffic analysis, and security. According to the dynamic nature of SDN, it needs a fast response due to the rapid changes in the network state. The SDN network topology can be modelled as a graph and stored in a graph database, and the traffic load of each switch is stored in the created graph. Consequently, a graph processing framework can be used to process the stored traffic data, and the results are utilized in traffic engineering to assist the SDN controller in network management. This paper provides a comprehensive literature survey involving graph techniques applied to SDN. Then, a summary of graph algorithms is presented. In addition, an overview of graph databases and graph processing frameworks is displayed. Finally, a model is suggested to integrate the graph database and graph processing framework in SDN traffic analysis.*

Index Terms— *Graph algorithms, Graph database, Graph processing frameworks, Network topology modelling, SDN, Traffic Analysis.*

I. INTRODUCTION

Today, crucial data challenges rely not only on classifying discrete data but they concentrate on relationships. Graph techniques prepare valuable tools for associated data to be utilized in financial markets, social networks, power grid networks, forecasting epidemic separation, communication systems, and traffic analysis in computer networks, to mention a few [1], [2]. Graphs boost search potentials in diverse information technology sectors, such as E-commerce sites for product recommendation and fraud detection. In addition, graph technologies and analytics are utilized to extract predictive features used in machine learning model building to fight economic crimes [3]. As data becomes more interconnected and systems become progressively vast and complicated, it's fundamental to leverage the priceless existence of relationships within data.

Lately, the huge amounts of information gathered, combined, and dynamically updated are enormous [4]. For Big Data storage and analysis, a graph database is promoted over a relational database for several reasons. First, it is difficult to model unstructured data adopting a relational database due to its rigid schema, making it inappropriate for handling similar data types. Contrarily, nodes and edges are used in a graph database to store and update information without confusing the current structure. Second, the search speed is faster multiple times than a relational database. Subsequently, scaling the graph database enhances the overall performance because search time decreases by providing more relationships in order to optimize the querying process [5].

In general, graphs consist of nodes or vertices, and the links between them are considered edges or relationships [6]. They are weighed as a powerful tool to envision and

Received 02/February/2023; Accepted 20/March/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.10

analyze highly connected data. Graphs are used constantly in networking, taking into account that modelling networks as graphs are natural. The networking devices (e.g., routers, switches, hosts, etc.) are represented as nodes, and the connections between these devices shape the edges [7]. Representing networks as graphs can assist in visualizing their deployment and recognizing interdependencies between these devices [8]. Moreover, graphs are used for traffic analysis to gain insights into device relationships. Finally, they can be used in service interruption, e.g., to find different routes when a device(s) or link(s) collapses to ensure service continuousness.

SDN is assumed to be influential as the next promising technology in the network softwarization sector [9]. SDN architecture consists of three planes: a data plane that includes the forwarding devices, a control plane in which the centralized controller is located, and an application plane where network control and management applications execute [10]. In SDN, the network control is separated from the infrastructure. All logic is transferred to the SDN controller, which is directly programmable [11]. Therefore, the network devices are only responsible for forwarding packets based on the controller's directions [12].

As a result, the SDN layered architecture can be utilized by modelling the network topology as a graph [10]. Traffic data are gained indirectly through the SDN controller and stored in a graph database. Graph algorithms are applied for traffic engineering tasks such as network planning and monitoring, routing, and load balancing using a graph processing framework.

This paper surveys and presents the application of graph techniques in networking. Popular graph algorithms that are applied in this sector are discussed. Then, the state-of-the-art graph databases are displayed and compared according to specific characteristics, followed by a demonstration of popular graph processing frameworks.

The rest of this paper is organized as follows: Section II displays the related works and a survey of several graph algorithms deployed in various computer networks. In section III, a brief explanation of some graph algorithms is given. Popular graph databases are detailed in section IV. In section V, well-known graph processing frameworks are displayed. Section VI presents a suggested traffic analysis model in SDN based on graph techniques. Finally, section VII concludes the paper.

II. RELATED WORK

This section discusses related research on graph techniques in networking, specifically in SDN networks.

The representation and maintenance of network topology information is the core of any network control and management system. In SDN, topology abstraction is used to view the network architecture from a different perspective. Authors in [13] suggested boosting the SDN network state by transforming a Network Markup Language (NML) semantic model into a graph imported into a graph database. They chose the Neo4j graph database because it supports property graphs and high-speed query processing compared to alternative graph databases.

A significant benefit of SDN architecture is the capability to use multiple controllers disseminated over the network to overcome the controller's single point of failure problem. In addition, it increases flexibility and scalability since it allows the deployment of a larger number of network switches. However, using multiple controllers increases the interconnection cost among the distributed controllers and the delay time between the

Received 02/February/2023; Accepted 20/March/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.10

controller and the switch. Authors in [14] proposed an approach based on the connected dominating set algorithm to decide the required number of SDN controllers and their locations to reduce the inter-controller delay time.

Since the widespread use of online applications and cloud administration, there has been a growth in network traffic [15]. As a result, numerous routing concerns are managed without increasing delay to ensure even traffic. SDN allows network characteristics and responses to be programmable. Authors in [16] address the issues caused by multi-path steering via SDN. The Depth-First Search (DFS) algorithms and breadth-first search (BFS) are illustrated to calculate the shortest path in the system. The most suitable search algorithm is determined by the round-trip time (RTT), with DFS being the preferred algorithm.

For data centers and service providers, accomplishing high network resiliency has been a priority for a long time. Although the centralized SDN view of the network may be beneficial, limited information is available about the behaviour of restoration algorithms and their recovery in a centralized controller when working in a worst-case scenario, such as a scaled wired mesh SDN topology. Authors in [17] reported the SDN implementation and analysis of various Minimum Spanning Tree (MST) algorithms, their essential performance indicators, and a preemptive technique for path restoration based on these findings. Results revealed significant time savings in the preliminary stages of constructing an MST.

SDNs have been effectively implemented in data centers and small to medium-sized businesses. On the other hand, adopting the SDN paradigm in Wide Area Networks (WAN) is more difficult because of several factors, including a higher likelihood of node and link failures and the lack of a devoted control channel. Authors in [18] presented a forwarding framework based on a Directed Acyclic Graph (DAG) that addresses the challenges of using SDN in sizable networks. The proposed framework mainly aims to reduce the number of entries needed on SDN switches, provide fast local restoration of single-node or link failures without the SDN controller's involvement, and avoid the potential of having inconsistent forwarding tables throughout updates. The DAG-based forwarding framework necessitates creating a DAG between each pair of ingress-egress switches and developing an index-based hashing algorithm to distribute the load throughout the DAG's pathways while avoiding TCP reordering difficulties.

Although graph techniques have been applied in various domains, no existing works provide a survey on graph applications in the networking domain. To fill this gap, a comprehensive study is provided in this paper of graph techniques applied to general networks, specifically SDN. Table I surveys some of the graph technique applications in computer networks.

TABLE I. GRAPH TECHNIQUES IN COMPUTER NETWORKS

Graph Algorithm/Concept	Ref	Issue	Network Type
Dominating Set	[19], [20]	Clustering	MANET
	[21]	Routing	MANET
	[22]	Clustering	WSN
	[1]	Distributed Clustering	Cloud Computing
	[9]	Controller Placement	SDN
	[23]	Lifetime Maximization	WSN
	[24]	Task Scheduling	Cloud Computing

Received 02/February/2023; Accepted 20/March/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.10

DAG	[25]	Network Reliability Improvement	WSN
Depth First Search	[18]	fast local restoration of single failures	SDN
	[16]	Quality of service Improvement	SDN
	[26]	Route Selection	WSN
Graph Coloring	[27]	Access Point Positioning	WSN
	[28]	Collision-Free MAC Protocol	USN
	[29]	Dynamic TDMA scheduling	MANET
Minimum Spanning Tree	[17]	path Restoration	SDN
	[31]	Route Selection	WSN
Shortest path	[32]	Routing	SDN
	[33]	Routing	USN

Acronyms: WSN: Wireless Sensor Network, MANET: Mobile Adhoc Network, SDN: Software Defined Networking, USN: Underwater Sensor Network

III. GRAPH ALGORITHMS

In this section, a number of graph algorithms that are deployed in communication networks are explained briefly.

- Depth-first search: In this algorithm, the search starts from a specific node considered a root. The exploration continues along every branch before backtracking. It finds a path between two nodes, discovers strongly connected components, topological sorting, and cycle detection [26].
- Shortest path: the shortest path from one node to another is a path in which the sum of the edge weights is the minimum. It is used in networking for routing [32].
- Minimum spanning tree: is a subset of the edges connecting all the nodes with the minimum sum of edge weights and containing no cycles. It is used in cluster analysis and route selection [31].
- Graph colouring: It assigns colours to the graph elements according to particular conditions. The graph vertices are coloured using N colours in vertex colouring, while any two adjacent vertices should not be coloured with the same colour. It is used in scheduling [27].

IV. GRAPH DATABASES

Recently, several new services become ineffective using relational databases, leading to the innovation of NoSQL databases like document-oriented and graph databases, which is considered a new trend [34]. A graph database is created to store, query, and modify network graphs. A network graph is a graphical representation of nodes and edges. [35]. Five prominent graph Databases, ArangoDB, Neo4j, OrientDB, Amazon Neptune, and Dgraph, are briefly illustrated as follows:

- ArangoDB: is an open-source, native multi-model database. It offers a flexible data model to store key-value pairs, documents, and graphs. Using ArangoDB, any data are accessed with a single declarative query language. It can scale horizontally and vertically in simple steps [36].
- Neo4j: is one of the widespread graph databases. It is written in Java language. Neo4j has a simple, flexible, and robust data model that easily adjusts to various applications

Received 02/February/2023; Accepted 20/March/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.10

[37]. It provides results depending on real-time data. One of the essential features of Neo4j is that it can work with REST API to operate with programming languages like Python, Java, and Scala [7], [38].

- OrientDB: is an open-source, NoSQL, multi-model database. It embeds and connects documents like a relational database [39]. OrientDB brings the power of graphs and the resilience of documents into an efficient and scalable database [40].
- Amazon Neptune: is a fast, fully managed, and reliable graph database. It makes it feasible to create and run applications working with highly connected data and auto-scaling storage for billions of relationships [41].
- Dgraph: is a GraphQL, distributed graph database. By minimizing network calls, Dgraph prioritizes concurrency in a distributed environment [42]. A comparison among the databases mentioned above is given in Table II.

TABLE II. COMPARISON OF VARIOUS GRAPH DATABASES

Feature	ArangoDB	Neo4j	OrientDB	Amazon Neptune	Dgraph
Database Model	Multi-model	Graph	Multi-model	Graph	Graph
License	Open Source	Open Source	Open Source	Commercial	Open Source
Implementation languag	C++	Java, Scala	Java	-	Go
Partitioning method	Sharding	Using Neo4j Fabric	Sharding	None	Yes
Replication	Supported	Supported	Supported	Supported	Supported
Transaction	ACID	ACID	ACID	ACID	ACID
Concurrency	Yes	Yes	Yes	Yes	Yes

A concise explanation of each feature listed in Table II. is provided as follows:

- Database Model: a Multi-model is a database capable of storing, indexing, and querying data in more than one model, such as relational, document, graph, and key-value.
- Partitioning is a generic term for dividing data across tables or databases. One specific type of partitioning is Sharding, in which the schema is typically replicated across multiple instances.
- Replication: techniques for storing data redundantly on numerous nodes.
- Transaction: transaction concepts maintain data integrity after non-atomic manipulations of data. ACID is an acronym for Atomicity, Consistency, Isolation, and Durability.
- Concurrency: assist in concurrent data manipulation.

V. GRAPH PROCESSING FRAMEWORKS

Processing large-scale graphs that cannot fit in a single machine memory is a new challenge. In systems like MapReduce, data processing doesn't perform well for large-scale graphs because these systems tend to partition the data and make the processing parallel. In graphs, however, data partitioning is not a simple task since every vertex in a graph is processed considering its neighbouring vertices. A possible replacement for the data-parallel model is the graph parallel processing model, in which a vertex-centric view of a graph is utilized. It shows good performance in large-scale graph processing [43]. Four

Received 02/February/2023; Accepted 20/March/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.10

graph processing frameworks, Pregel, GraphLab, Giraph, and GraphX, are explained as follows:

- **Pregel:** was created by Google and influenced by the bulk synchronous parallel (BSP) processing model. It is a data flow system to ease big graph processing in which the number of nodes can be billions. It is used to solve complex problems expressed using the MapReduce model. Pregel's execution of applications is a sequence of iterations known as supersteps. A vertex in a superstep receives all messages delivered to it in the previous superstep, updates its local state, and sends messages to its adjacent neighbours to be transferred to the next superstep [44].
- **GraphLab:** while Pregel facilitates big graph processing, it is restricted by its firm synchronization mechanism. GraphLab uses an asynchronous model where vertices are directly read and update data in their scope rather than sending read/update requests using message passing [45].
- **Apache Giraph:** it is a real-time, iterative graph processing system. It is used in social media networks to find connections between entities. Facebook and Twitter have chosen it because of its high scalability [46]. Giraph, like its counterpart Pregel, is inspired by the BSP model. However, Giraph enhances multiple features beyond Pregel, including edge-oriented input, master computation, and more. Since it continues to develop steadily, Giraph has become a natural choice to release the potential of a structured dataset at a large scale [47].
- **GraphX:** it is built on top of Spark. It combines graph-parallel and data-parallel models in one system with a single API. GraphX extends Spark RDD (Resilient Distributed Dataset) by establishing a directed multigraph with properties tied to each vertex and edge [48]. It lets users view data as graphs and collections (RDDs) without data movement or replication. GraphX has a set of operators (e.g., subgraph, joinVertices) to support graph processing. In addition, a collection of graph algorithms such as PageRank and strongly connected components is used in graph analysis [49]. A comparison among the frameworks mentioned above is shown in Table III.

TABLE III. GRAPH PROCESSING FRAMEWORKS COMPARISON

Feature	Pregel	GraphLab	Giraph	GraphX
Programming Model	Vertex-centric	Vertex-centric	Vertex-centric	Edge-centric
Partitioning	Edge-cut	Edge-cut	Edge-cut	Vertex-cut
Architecture	Distributed	Single machine	Distributed	Distributed
Computational Model	BSP	N/A	BSP	GAS
Communication Model	Message Passing	Shared memory	Message Passing	Dataflow
Coordination	Synchronous	Asynchronous	Synchronous	Synchronous
Storage	Disk-based	Disk-based	Disk-based	Memory-based

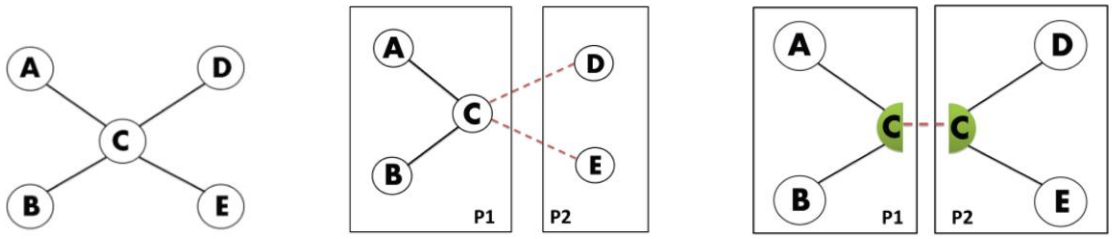
A brief explanation of each feature in Table III is provided as follows:

- Programming Model: two programming models are mentioned above; Vertex-centric, a.k.a Edge-Cut, and Edge-centric, a.k.a Vertex-cut. The most mature model is Vertex-centric, in which the graph is partitioned according to its vertices, and the vertices are distributed over various partitions. While in the Edge-Centric model, edges are the basic

Received 02/February/2023; Accepted 20/March/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.10

unit of calculation and partitioning. Vertices connected to edges residing in several partitions are duplicated and shared among these partitions. Therefore, each edge is allocated to one partition, whereas each vertex might reside in multiple partitions. The two programming models are shown in *Fig. 1*.



a. A sample graph b. Vertex-centric (edge-cut) c. Edge-centric (vertex-cut)

FIG. 1. PROGRAMMING MODELS IN GRAPH PROCESSING FRAMEWORKS [8].

- Partitioning: Graph processing frameworks are classified into three architectural models; Distributed, Shared-Memory, and Heterogeneous. The distributed architecture consists of numerous processing units, and every unit has only access to its private memory. In Shared-Memory architecture, a single machine contains one processing unit with at least one CPU core, and physical memory is shared across all the cores. While in a Heterogeneous architecture, not all processing units are evenly powerful.
- Computational Models: graph processing frameworks are divided into two-phase and three-phase models. The most illustrative model in the category of the three-phase model is Bulk Synchronous Parallel (BSP), a parallel programming model used in numerous graph processing frameworks. BSP processing encompasses a series of supersteps with a synchronization barrier. Another three-phase computational model is Gather-Apply-Scatter (GAS). In the gather phase, data on adjacent nodes and edges are retrieved and gathered by performing a generic summing of all-inclusive neighbouring vertices and edges of a vertex. The apply phase is user-defined. It might range from a numerical summation to data accumulation over all adjacent edges and vertices. The outcomes from the gather phase are employed to modify the central vertex values in the apply phase. Eventually, in the scatter phase, the central vertex's recent data revive the values on neighbouring edges [50].
- Communication Model: Graph processing frameworks communicate with their vertices, edges, and partitions using a variety of approaches. In the Message Passing approach, communication is performed by transmitting messages from a single entity (vertex, edge, or an element in a local or remote partition) to another in the graph. Whereas using Shared Memory, numerous processing modules can access the memory location simultaneously, involving read and write to that location.
- Coordination: Current graph processing systems can use various distributed coordination. In the Synchronous model, simultaneous workers repeatedly process their portion of the work through well-defined and globally coordinated iterations. While in the Asynchronous model, the execution model has no global barriers. Therefore, a new execution phase will begin immediately after a worker's current iteration completes its calculation.
- Storage: there are two common approaches to storing graph data; disk-based and memory-based. Initially, when loading the graph, a disk-based execution gets the graph data from physical disks, and during the execution of writing and reading portions of the

Received 02/February/2023; Accepted 20/March/2023

DOI: https://doi.org/10.33103/uot.ijccce.23.3.10

graph state to and from the disk. While in the memory-based, the graph and its states are solely stored in memory at runtime for storing and analyzing massive amounts of data.

VI. THE SUGGESTED MODEL FOR TRAFFIC ANALYSIS IN SDN

A graph database and a graph processing framework could work seamlessly for traffic analysis in SDN. The cooperation between these tools is used to compute the traffic load distribution of every switch in the SDN data plane in a specific period, and each switch is ranked based on its traffic load. These findings can detect network congestions and provide network design guidelines., e.g., increasing/decreasing switch capacity and detecting underutilized links in certain parts of the network. The suggested model for traffic analysis in SDN utilizing these tools is shown in *Fig. 2*.

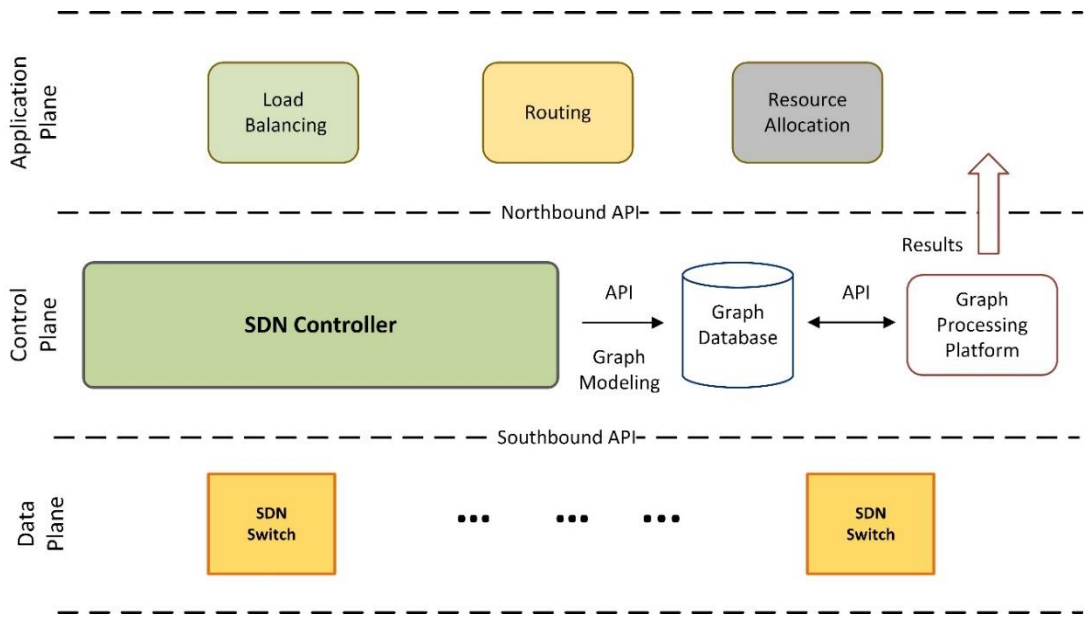


FIG. 2. THE SUGGESTED MODEL FOR SDN TRAFFIC ANALYSIS.

VII. CONCLUSIONS

Several graph techniques utilized in traffic analysis in SDN are introduced in this work. First, graph algorithms with their applications in various computer networks are presented. Then, multiple graph databases that are widely used are explained. Following that, popular graph processing frameworks with their characteristics are discussed. Finally, a model for traffic analysis in SDN is suggested. The SDN network topology can be modelled as a graph to facilitate storing the traffic data for each networking device in a graph database to be processed by a graph processing framework. According to the SDN architecture, the suggested model is implemented in the SDN control plane since it requires high interaction with the underlying network and to respond quickly to the network events. The selection of the graph database depends on two factors. First is the capability of the graph database to cooperate with the selected SDN controller to receive and store the collected traffic statistics. Second, its flexible integration with the selected graph processing framework to build an efficient SDN traffic analysis model.

Received 02/February/2023; Accepted 20/March/2023

ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.10

REFERENCES

[1] K. D. Rangaswamy and M. Gurusamy, "Application of Graph Theory Concepts in Computer Networks and its Suitability for the Resource Provisioning Issues in Cloud Computing-A Review," Journal of Computer Science, 2018.

[2] B. Kan, W. Zhu, G. Liu, X. Chen, D. Shi, and W. Yu, "Topology Modeling and Analysis of a Power Grid Network Using a Graph Database," International Journal of Computational Intelligence Systems, 2017.

[3] J. Webb, F. Docemmilli, and M. Bonin, "Graph Theory Applications in Network Security," SSRN Electronic Journal, 2015.

[4] M. Needham and A. E. Hodler, "Graph Algorithms Practical Examples in Apache Spark & Neo4j", 1st edition, O'Reilly, 2019.

[5] B. Ristevski, "Using Graph Databases for Querying and Network Analysing," International Conference on Applied Internet and Information Technologies, Zrenjanin, Republic of Serbia, 2019.

[6] Patil N. S., Kiran P, Kavya N. P., and Naresh Patel K. M., "A Survey on Graph Database Management Techniques for Huge Unstructured Data," International Journal of Electrical and Computer Engineering (IJECE), Vol. 8, No. 2, April 2018.

[7] J. Guia, V. G. Soares, J. Bernardino, "Graph Databases: Neo4j Analysis", In Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017).

[8] S. Heidari, Y. Simmhan, R. N. Calheiros, and R. Buyya, "Scalable Graph Processing Frameworks: A Taxonomy and Open Challenges," ACM Computing Surveys, vol. 51, no. 3, article 60, 2018.

[9] A. S. Dawood and M. N. Abdullah, "Adaptive Performance Evaluation for SDN Based on the Statistical and Evolutionary Algorithms," Iraqi Journal of Computers, Communications, Control & Systems Engineering (IJCCCE), 2019.

[10] J. Taheri, "Big Data and Software Defined Networks," 1st edition, The Institution of Engineering and Technology, London, United Kingdom, 2018.

[11] W. Queiroz, M. Capretz, and M. Dantas, "A MapReduce Approach for Traffic Matrix Estimation in SDN," in IEEE Access, vol. 8, 2020.

[12] R. S. Al mahdawi and H. M. Salih, "Optimization of open flow controller placement in software defined networks," International Journal of Electrical and Computer Engineering (IJECE), Vol. 11, No. 4, August 2021.

[13] T. D. P. C. D. Souza, C. E. Rothenberg, M. A. S. Santos, and L. B. D. Paula, "Towards Semantic Network Models via Graph Databases for SDN Applications," 2015 Fourth European Workshop on Software Defined Networks, 2015.

[14] A. Alowaa and T. Fevensa, "Towards Minimum Inter-Controller Delay Time in Software Defined Networking," The 15th International Conference on Future Networks and Communications (FNC), 2020.

[15] M. Abbasi, H. Rezaei, V. G. Menon, L. Qi, and M. R. Khosravi, "Enhancing the Performance of Flow Classification in SDN-Based Intelligent Vehicular Networks," IEEE Transactions on Intelligent Transportation Systems, 2020.

[16] F. M Shamrat, P. Ghosh, Md. Sultan, A. Majumder, and I. Mahmud, "Software-Defined Networking with Multiple Routing Utilizing DFS to Improving QoS," 2nd International Conference On Emerging Technologies In Data Mining And Information Security, 2021.

[17] W. Collymore, "A Preemptive Hybrid Approach to Minimum Spanning Tree Restoration in Large Mesh SDN Networks," 7th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), 2019.

[18] S. Avallone and U. Ashraf, "A DAG-Based Forwarding Paradigm for Large Scale Software Defined Networks," IEEE Transactions on Network and Service Management, 2020.

[19] T. N.Tran, T. Nguyen, and B. An, "An Efficient Connected Dominating Set Clustering Based Routing Protocol with Dynamic Channel Selection in Cognitive Mobile Ad Hoc Networks," Electronics, 2019.

[20] A. K. Das and B. Kalita, "Dominating Set Based Clustering Algorithm for Mobile Adhoc Network," International Journal of Recent Technology and Engineering (IJRTE), 2019.

[21] K.G. Preetha and A. Unnikrishnan, "Enhanced domination set based routing in mobile ad hoc networks with reliable nodes," Computers and Electrical Engineering, 2017.

[22] J. Pan, L. Kong, T. Sung, P. Tsai, and Václav Snášel, "A Clustering Scheme for Wireless Sensor Networks Based on Genetic Algorithm and Dominating Set," Journal of Internet Technology, 2018.

[23] T. Pino, S. Choudhury, and F. Al-Turjman, "Dominating Set Algorithms for Wireless Sensor Networks Survivability," IEEE Access, 2018.

[24] X. Shao and Z. Xie, "A scheduling algorithm for applications in a cloud computing system with communication changes," Expert Systems, 2018.

[25] E. Al-hawri, N. Correia, and A. Barradas, "DAG-Coder: Directed Acyclic Graph Based Network Coding for Reliable Wireless Sensor Networks," IEEE Access, 2016.

[26] J. Lyu, Y. Ren, Z. Abbas, B. Zhang, "Reliable Route Selection for Wireless Sensor Networks with Connection Failure Uncertainties," Sensors, 2021.

Received 02/February/2023; Accepted 20/March/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.10

[27] X. Zhu, Y. Mao, J. Qi, and R. Wang, "An AP Graph Coloring Deployment Algorithm for Indoor Sensor Network Positioning," Ad Hoc & Sensor Wireless Networks, 2021.
[28] F. A. Alfouzan, A. Shahrabi, S. M. Ghoreyshi, and T. Boutaleb, "A Collision-Free Graph Coloring MAC Protocol for Underwater Sensor Networks," IEEE Access, 2019.
[29] Y. Ye, X. Zhang, L. Xie, and K. Qin, "A Dynamic TDMA Scheduling Strategy for MANETs Based on Service Priority," Sensors, 2020.
[30] J. Anzola, J. Pascual, and G. Tarazona, "A Clustering WSN Routing Protocol Based on k-d Tree Algorithm," Sensors, 2018.
[31] J. Lyu, Y. Ren, Z. Abbas, B. Zhang, "Reliable Route Selection for Wireless Sensor Networks with Connection Failure Uncertainties," Sensors, 2021.
[32] M. Nowak, P. Pecka, "Routing Algorithms Simulation for Self-Aware SDN," Electronics, 2022.
[33] R. Gomathi and J. Manickam, "Energy Efficient Shortest Path Routing Protocol for Underwater Acoustic Wireless Sensor Network," Wireless Pers Commun, 2017.
[34] S. Jouili and V. Vansteenberghe, "An Empirical Comparison of Graph Databases," 2013 International Conference on Social Computing, 2013.
[35] Mark Needham and Amy E. Hodler, "Graph Algorithms Practical Examples in Apache Spark & Neo4j", O'Reilly, 2019.
[36] K. Mavrogiorgos, A. Kiourtis, A. Mavrogiorgou, and D. Kyriazis, "A Comparative Study of MongoDB, ArangoDB and CouchDB for Big Data Storage", In Proceedings of the 2021 5th International Conference on Cloud and Big Data Computing (ICCBDC '21), Association for Computing Machinery, New York, NY, USA, 8–14.
[37] Q. Ruhimat, G. Fajariyanto, D. Firmansyah, and Slamin, "Optimal computer network based on graph topology model," IOP Conf. Series: Journal of Physics: Conf. Series 1211, 2019.
[38] F. López and E. Santos De La Cruz, "Literature review about Neo4j graph database as a feasible alternative for replacing RDBMS", Industrial Data, 2015.
[39] D. Fernandes and Jorge Bernardino, "Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB", In Proceedings of the 7th International Conference on Data Science, Technology and Applications (DATA 2018).
[40] D. Fernandes and J. Bernardino, "Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB", Proceedings of the 7th International Conference on Data Science, Technology and Applications (DATA 2018), 2018.
[41] B. Bebee, R. Chander, A. Gupta, A. Khandelwal, S. Mallidi, M Schmidt, R. Sharda, B. Thompson, and P. Upadhyay, "Enabling an Enterprise Data Management Ecosystem using Change Data Capture with Amazon Neptune", In ISWC (Satellites), 2019.
[42] Dgraph, [Online]. Available: https://dgraph.io/.
[43] P. Zecevic and Marko Bonaci, "Spark in Action," Manning Publications, 2017.
[44] K. Ammar and M. Tamer Ozsu, "Experimental Analysis of Distributed Graph Systems," Proceedings of the VLDB Endowment, 2018.
[45] B. Chambers and M. Zaharia, "Spark: The Definitive Guide Big Data Processing Made Simple," O'Reilly, 2018.
[46] A. Mohan and Remya G, "A Review on Large Scale Graph Processing Using Big Data Based Parallel Programming Models," IJ Intelligent Systems and Applications, Feb 2017.
[47] S. Sakr, F. Orakzai, I. Abdelaziz, and Z. Khayyat, "Large-scale graph processing using Apache Giraph", Cham: Springer International Publishing; 2016.
[48] Michael S. Malak and Robin East, "Spark GraphX in Action," Manning Publications, 2016.
[49] GraphX, [Online]. Available: https://spark.apache.org/graphx/.
[50] J. Koch, C. L. Staudt, M. Vogel, and H. Meyerhenke, "An Empirical Comparison of Big Graph Frameworks in the Context of Network Analysis," Social Network Analysis and Mining, 2016.

Received 02/February/2023; Accepted 20/March/2023
 ISSN (Print) 1811-9212 *ISSN (Online) 2617-3352*