Vol. 1 No. 1 (2023) pp. 1-53: Journal of Intelligent Systems and applied data science (JISADS)

ISSN (2974-9840) Online

We are pleased to publish the inaugural issue of the Journal of Intelligent Systems and Applied Data Science (JISADS). JISADS is a multidisciplinary peer-reviewed journal that aims to publish high-quality research papers on Intelligent Systems and Applied Data Science. Published: **2023-07-15** and the issue closed on **5 articles** and total pages are **53 pages**.

Editor-In-Chief:
Wasim Ali
Politecnico di Bari, Italy
Editor@jisads.com

Journal of Intelligent System and Applied Data Science (JISADS)

Journal homepage : https://www.jisads.com

ISSN (2974-9840) Online

Impact Of Using Fiber Delay Scheme On Burst Loss Ratio And Delay Using Offset Time Algorithm For Optical Burst Switching Networks

Laila A. Wahab Abdullah Naji[1], *Ibrahim Khider Eltahir*[1*], *Hadeil Haydar Ahmed Elsheikh*[1]

Tefke2010@Gmail.Com, Ibrahim_Khider@Hotmail.Com, Hdola1989rm@Gmail.Com.

[1] University of Aden-Faculty of Aden, Yemen

Abstract

The optical burst switching (OBS) paradigm is an intermediate optical switching solution between optical packet switching (OPS) and optical circuit switching (OCS). In addition, OBS has enormous bandwidths that can satisfy the requirements of bandwidth applications and the growing number of end users. OBS, suffer from burst contention due to a lack of optical buffers. This problem results in a high burst loss ratio and increased end-to-end delay, thus degrading the performance of the OBS network. This study has proposed a Fuzzy Offset Time algorithm (FOTA) to address the above issues. The fuzzy input comprises three parameters: B.Size, Distance, and Q.Delay. In this study, Five defuzzification techniques are used Centroid, bisector, largest of maximum, smallest of maximum, and mean of maximum (CM00, BM04, LM02, SM03, MM01, respectively) applying to both maximum and algebraic sum accumulation techniques using fiber delay schemes. The results of FOTA show the defuzzification (LM02 and LS02) have effects in reducing BLR (burst loss ratio) while the defuzzification (SM03 and LS02) have effects in reducing end-2-end delayed, respectively.

Keywords: OBS Networks, FOTA, Fuzzy Logic controller, average e-to-end delay, BLR

1. INTRODUCTION

In the latest years, the interest in the internet has been growing. Still, today, users rely heavily on the Internet of Things (IoT), artificial intelligence (AI), multimedia applications, and other internet technologies such as marketing and banking online. These technical advancements need a large amount of bandwidth to be implemented. The optical fiber may be offered to solve to match the huge requirement of raw bandwidth. A single optical fiber can give a bandwidth of up to 50 THz, so wavelength division multiplexed (WDM) is one solution that matches the requirement of huge raw bandwidth [1]. WDM technology is widely used to meet the significant increase in the demand for channel capacity due to the rising customer and to face the challenges [2] and gives a large amount of bandwidth [3]. Optical burst switching allows dynamic sub-wavelength data switching, eliminating throughput constraints and maximizing bandwidth utilization. Different user data types are merged at the OBS network's edge node before being sent as data bursts. Every burst has a control packet with its information in it. A separate control channel has been designated for the transmission of this packet. Due to its smaller size, this control packet can contain information for hundreds of data channels. At each intermediate OBS node, the control packet undergoes an O/E/O conversion and is

electrically switched to obtain a configuration with the switch. The network is establishing an offset time. Before the core can allocate resources to the upcoming burst, offset time is the amount of time it takes to process the information in the control packet; it is referred to as the processing configuration delay. The data burst can immediately switch in an optical domain with the appropriate offset time. At the intermediate nodes, optical RAMs or FDLs (Fiber delay lines) will be less necessary due to this.[4]

OBS is an attractive and preferable choice over Optical Packet Switching (OPS) and Optical Circuit Switching (OCS), as it can handle the dramatic increase in multimedia applications' traffic [2]. Unlike OCS and OPS, data is transmitted in bursts instead of packets. The bursts are grouped and sent based on their destination. Table1: gives a comparison between the three-switching technology. [5][6][7]

Table 1: Comparison of the three techniques of optical switching

Characteristics / Properties	Bandwidth	Set-up	Optical buffer	Overhead Processing	Traffic Adaptability	Speed's Switching	Complexity Processing	Signalling Scheme
Circuit	Low	High	Unwanted	Low	Low	Slow	Low	Two ways
Packet	High	Low	Wanted	High	High	Fast	High	One Way
Burst	High	Low	Unwanted	Low	Low	Moderate	Medium	One way

OBS network architecture consists of edge and core nodes (ingress and egress nodes). All IP packets from different access networks are aggregated in the form of bursts by the ingress node[8]. See Figure1 [9].

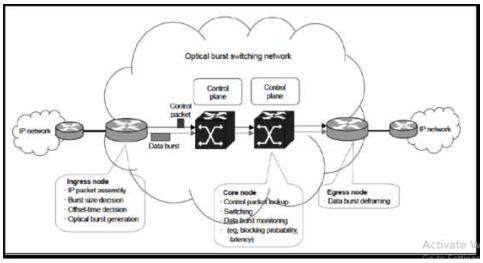

Figure 1: OBS network architecture

The main functions of an ingress node are: aggregating the bursts, generation of the burst header packet (BHP), determination of the offset time, determining the routing and wavelength assignment (RWA), wavelength reservation, and signaling. The signaling process is facilitated by combining the packets into a single data burst, which reduces the number of requests at the core nodes [9]. Also, the main processing of the core nodes is, switching all-optical data bursts from one input port to another depending on the information in the BHP. The core node resolves contention between bursts by deciding the routing of the burst. The egress node's main functions are to disassemble the large bursts to the original packet and route them to their respective final destinations forwarding [10].

The following structure is used for this paper: Section II discusses the related works on offset time. The proposed Algorithm of Offset Time is reported in section III. The results' simulations are explained in section IV, and the conclusion of this study is in section V.

2. RELATED WORKS OF OFFSET TIME ALGORITHM

The time data bursts following its control packet after some time is known as offset time. The offset time allows the switch to handle the control packet. This includes getting the needed resources and setting up the optical switch at transitional OBS nodes so that the next burst can pass through each transitional OBS node without waiting for the resources or the switching fabric. The offset time is set to complete all these operations before the data burst arrives. Different offset times can isolate different traffic classes, allowing for service differentiation. [11].

The offset time should be longer than the total BHP processing times at all nodes because of the configuration and reservation time spent at the core nodes. The burst will be discarded if the offset time is shorter than the processing time because it will arrive before the BHP. Thus, the burst loss probability and end-to-end delay are used to evaluate the efficiency of

any offset-time method [9].Figure2 displays an OT scheme in OBS networks [12].

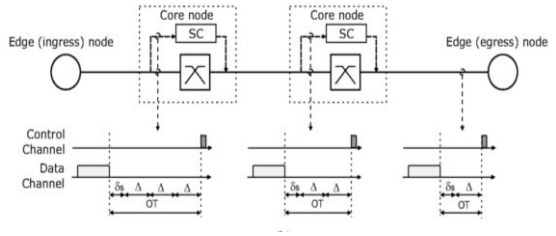

Figure 2 Displays an OT scheme in OBS networks.

Due to the processing delay of the CP at all nodes from ingress to egress, it is challenging to have all bursts with the same offset time [13]. The Virtual Fixed Offset Time (VFO) was presented as a solution to the issue of processing delays that vary from one to another. VFO processes bursts in accordance with the burst arrival time rather than the CP arrival time. The burst with the earliest arrival time is scheduled after the CPs have been sorted by their burst arrival time. However, fiber delay lines (FDLs) utilize the burst offset time at each node to delay bursts and ensure that no other bursts arrive before the earliest burst scheduled. The VFO scheme sends the smaller bursts directly but increases the delay time for larger bursts; this operation causes the problem of unfairness between bursts, even though it delays the larger bursts in each node using FDLs beside this, it is the using of FDLs is costly.

Fixed OT scheme is derived from [14] the just-enough-time signaling protocol (JET). OT for JET protocol is constant and is the summation of processing times for all hops, and the time of switching configuration, switching time, and processing times at each node are equal. Estimation OT must know how many hops there are between the source and destination nodes. Due to waiting delays in the control channel, these times may vary between nodes. Having a defined offset time has the disadvantage of allowing small bursts to be sent sooner and without delay. In contrast, there won't be enough time to send large bursts.

[15], also proposed an algorithm that maximizes resource use while reducing loss. The suggested algorithm requires wavelength of full conversion capabilities from the nodes whose routing information is provided through nodes in the OBS Networks. The edge nodes choose the best routes based on the typical link availability. Updates are made to the average traffic volume and link availability. Although burst loss results were more significant than utilization outcomes under low-load conditions, this technique outperformed the other examined algorithms.

To achieve a high degree of isolation between bursts of varying sizes, the Adaptive offset-time scheme gives larger bursts more offset time [16]. Additionally, if the isolation of OT is equal to the size of the burst, a degree of separation of one can be attained. As a result, the network's overall performance is enhanced, and the blocking probability is decreased when the additional OT is applied to a larger burst. The scheme's drawback is that the additional offset time will result in long delays and higher loss penalties. Furthermore, the adaptive OT system is better appropriate for usage in long-distance networks with high real-time traffic, making the offset time insignificant in comparison to the delay of transmission.

In the Jacobson-Karels algorithm, when the period of burst assembly is less than the OT, this method reduces the OT by transmitting the control packet containing an estimate of the burst length just before the period of assembling the burst expires. This method transmits bursts faster than the conventional approach and does not add additional offset time delay. The retransmission time (in the transmission control protocol TCP) should be calculated using this algorithm, and the burst length should be predicted and then added in the header of the BHP, so BHP can be sent before the time of burst assembly expires. Inaccurate estimation causes an increase in burst loss17].

A method is provided by [18] for obtaining a moderate OT that fits the insufficient OT drop ratio criteria while preserving a tradeoff between the two. The burst loss probability, which is used as a monitoring variable, is used to allocate offset time dynamically to reach this equilibrium. After measuring the background traffic on the core nodes, the OT is dynamically set.

The burst scheduling techniques published in the previous tenses, in contrast to conventional scheduling algorithms, are focused on maximizing the utilization of local networks. For instance, [19] proposes a method that uses local networks without increasing the burst loss rate. The gap between bursts is reduced by connecting upcoming bursts with existing bursts. A variable offset-time value can be achieved by establishing minimum and maximum OT limits instead of defining an offset-time value. Alternately, the bursts can be aligned at the beginning or end of the selected vacuum. However, no evidence exists that this method changes offset time's value.

The authors [20] examined how offset time affected the burst loss ratio. As the adjusting parameter, OT was used. The researchers suggested controlling the closed-loop feedback method for an adaptive offset time. As a result of the feedback that was received, the offset time is changed adaptively. The model supplies the BHP using the shortest offset time value before its associated burst.

In [21], the intelligent offset time is better than conventional and adaptive offset time in terms of E2E delay and BLR. The authors used in this algorithm a fuzzylite program and Omnet++ simulation to perform this algorithm using centroid defuzzification to evaluate these two matrices.

3. THE PROPOSED ALGORITHM OF OFFSET TIME

The contention is the main problem in the core network of OBS networks which causes drop bursts; so many researchers try to avoid contention at the edge node by using different techniques to minimize contention. Controlling the offset time delay has been proposed in this study, so the size of the burst, the time of assembled burst wait at the assembler, and the distance from ingress to the egress node is the main parameter in this study to minimize the contention. In this proposed algorithm, the main input parameters to the fuzzy logic controller are B. Size, Q.Delay (the time spent by burst in queuing before being directed to the core nodes), and distance (hops number), while OT is the output control variable. The new output value is a Fuzzy offset time (FOT) used for the burst header packet BHP to reserve the resources (wavelengths) needed for successful transmission. This algorithm is multi-input single output and has 27 rules to evaluate it.

The design of FOT algorithms consists of two main components:

1. Design the fuzzy logic controller of fuzzy Offset Time, where its component is:
    - Identification of control variables: The control variables used in this algorithm to generate the Fuzzy Offset Time are the Burst Size, Q. Delay, and distance. These control variables are used as inputs to the FLC. The last control variable is Offset Time, the output parameter where the three inputs are used to calculate the adaptive value of offset time.
    - Fuzzification of control variables: Here, the input and output control variables are converted into fuzzy forms using triangular membership functions (TMF).
    - Knowledge base formation: In this stage, a set of rules are formulated by the Fuzzy Logic Controller (FLC) where four (three inputs and one output) sets of fuzzy rules are defined for FLC.
    - Fuzzy Inference Engine formation: a Mamdani fuzzy inference engine was chosen.
    - Defuzzification of Fuzzy Output variables: The output control variable is in the form of fuzzy to convert into its crisp value. In this phase, a defuzzification process produced an output that achieves the objective of this study.
2. Designing the algorithm and integrating it with the FLC: Here, the FLC and the FOTA procedure are integrated to explain the FOT algorithm in this stage.

The OBS paradigm was simulated on the Omnet++ simulation framework version 4.2.2 platform. Omnet++ was chosen for this study due to its many useful features. A few of its beneficial attributes are open source, free for academic research,

and ease of programming due to object-oriented programming basics. Omnet++ simulation framework was selected because it has a well-developed OBS simulation model (component or plug-in).

In this study, each control variable is divided into three partitions, each witha label name, as shown in Table2. A triangular membership function is used for every partition, which is the final stage of the fuzzification process.

Table 2: Fuzzy input variable with operation range

	Fuzzy variables	Universe of Discours	Partition label (T)
Input	B. Size	[0, 60] k bytes	Small
			Medium
			Big
	Distance	[0, 8]	Short
			Middle
			Long
	Q. Delay	[0, 400] µs	Low
			Average
			High
Output	FOT	[0, 100] µs	Little
			Moderate
			Large

In this phase, the simulation results obtained from the experiment were analyzed, assessed, and discussed. Burst loss ratio (BLR) and burst end-to-end delay, which are the two-performance metrics, were used to assess all of the study's finding results in this study. They were chosen to ensure the algorithms are appropriate for BLR and end-to-end delay-sensitive applications. More important, the results of the analysis and evaluation are as the following:

Proposed Fuzzy Offset Time algorithm versus the existing intelligent offset time algorithm:

A. Applying different defuzzification processes such as Bisector, Largest of Maximum, Smallest of Maximum, and Mean of Maximum "with FDL" versus Centroid using maximum as the aggregation type.

B. Applying different defuzzification processes such as Bisector, Largest of Maximum, Smallest of Maximum, and Mean of Maximum "with FDL" versus Centroid using Algebraic sum as the aggregation type.

The parameters used for the Omnet++ simulation framework's OBS modules plug-in is shown in Table 3

Table 3: Parameters and setting of OBS Simulation

Parameter	Value
Network	NSFNET
Number of	4 (3 data
Bandwidth	1
Packet Size	1250
Control	10
Propagation	1
Packet	Exponential
Scheduling	LAUC
Timeout (s)	0.0005
Burst	1.5K
Burst	60K
Load (min)	0.1
Load (max)	1
Load	0.1
Signalling	JET
Optical	ON

The network is NSFNET, consisting of 14 bidirectional links, uniform traffic distributed across all source/destination pairs, and one wavelength allocated as a control packet channel on every link. The algorithm design compares with the Intelligent offset Time (IOT). The two important evaluation metrics in this study are BLR and end-to-end delay.

4. RESULT AND DISCUSSION

From Figure 3, it is clear that there is an increase in the burst loss ratio for all defuzzification methods when FDLs are employed as optical buffers. FOT LM02 is the best in the case of BLR due to the low burst loss ratio exhibited by FOT LM02, which is due to its ability to use its rules in the fuzzy logic controller to produce an adequate value of offset time between the BHP and data burst of suitable sizes. As a

result, FOT LM 02 generates bursts of large sizes that lower the network's level of congestion and hence minimize burst contention and their loss. However, the FOT LM02 configuration provides better network performance than other configurations under heavy load.

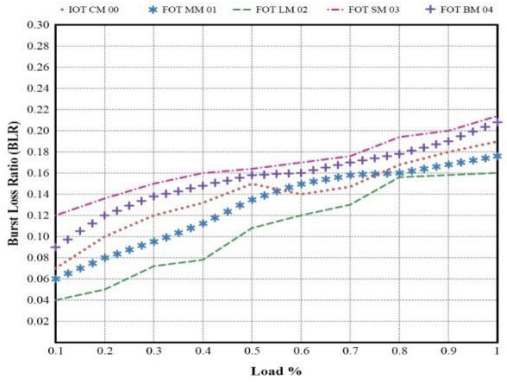

Figure 3: Burst Loss Ratio versus offer load for diffèrent défuzzification techniques and Maximum aggregation method with FDLs.

As shown in Figure 4, FOT SM03 has better performance in terms of end-to-end delay than the three configurations of the FOT algorithm and IOT algorithm due to the small burst size. However, despite the heavy traffic load and growing burst size, FOT SM03 has a constant average delay value from 0.8 to 1.0. FOT LM02 has the highest delay due to the large burst generation, which tends to have a longer transmission time, although, of this, it has less burst loss ratio. When the burst size is small, BHP does not need a large processing time and is directed to the destination without any loss and will not take time in buffering during processing BHP. Unlike when the burst size is large, it needs buffering for a fixed and predetermined duration which is limited by fiber length, so when there is a large burst in queuing, this causes the burst to drop and hence increases the BLR because the FDL process a FIFO system.

FOT SM03 has a high burst loss ratio because it generates many large bursts, which causes high contention. In return, it has the best network performance because it produces less end-to-end delay ratio due to generating small bursts.

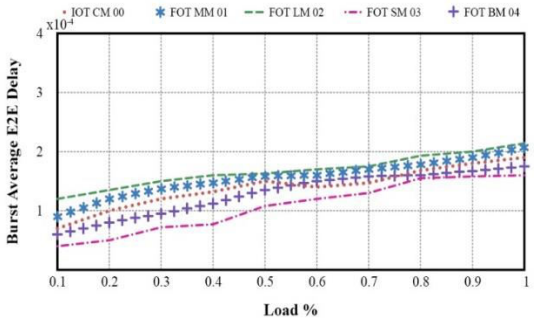

Figure 4: Burst average end-to-end delay versus offer Load for different defuzzification techniques and Maximum aggregation method with FDLs

Figure 5 shows an increase in the burst loss ratio for all FOT LS02 while IOT CS00 still without change on both burst loss ratio and end-to-end delay means; IOT CS00 has no effect while changing the aggregation from maximum to algebraic sum. From 0.8 to 1.0, it is noticeable that IOT CS00 and FOT MS01 have the same end-to-end delay due to generating equal data burst size and causingan equal burst loss ratio.FOT LS02 displayed the best burst loss ratio when compared with other defuzzification techniques, and in return. At the same time, IOT CS00 is better than both FOT MS01 FOT BS04 and FOT SS03, as FOT LM 02 generates bursts of large sizes that lower the network's level of congestion, so it minimizes burst contention and reduce loss of burst. However, the FOT LM02 configuration provides better network performance than other configurations under heavy load. However, the fuzzy rule the configuration FOT LS02 uses effectively reduces the burst loss ratio at all offered loads.

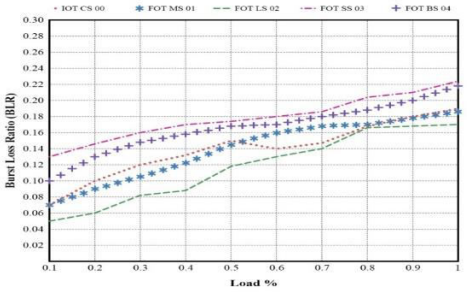

Figure 5: Burst Loss Ratio versus offer load for different defuzzification techniques And Sum aggregation method with FDLs.

As shown in Figure 6, FOT SM03 has better performance in terms of end-to-end delay than the three configurations of the FOT algorithm and IOT algorithm due to the small burst size. However, despite the heavy traffic load and growing burst size. From 0.1 to 0.4, IOT CM00 is better than FOT MM01 and FOT LM02 regarding end-to-end delay. FOT LM02 has the highest delay due to the large burst generation, which tends to have a longer transmission time, although, of this, it has less burst loss ratio. When the burst size is small, BHP does not need a large processing time and is directed to the destination without any loss and will not take time in buffering during processing BHP. Unlike when the burst size is large, it needs buffering for a fixed and predetermined duration which is limited by fiber length, so when there is a large burst in queuing, this causes the burst to drop and hence increases the BLR because the FDL process a FIFO system. FOT SM03 has a high burst loss ratio because it generates a high number of busts which causes high contention. In return, it has the best network performance because it produces less end-to-end delay ratio due to generating small bursts.

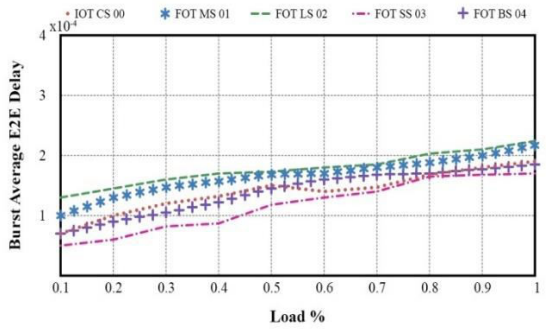

Figure 6: Burst average end-to-end delay versus offer Load for different defuzzification techniques and Sum aggregation method with FDLs.

5. CONCLUSION

This study presents an algorithm using the fuzzy offset time (FOTA) to reduce the burst loss probability (BLR) and end-to-end delay in optical burst switching networks. Adding FDLs increases the burst loss ratio, as in Figures 3 and 5; in contrast, end-to-end delay performs the best in Figures 4 and 6. Compared to different defuzzification, FOT LM02 performs the best loss ratio, while FOT SM03 performs the best on end-to-end delay. In future research, it is recommended to explore alternative aggregation methods beyond the ones utilized in this study, such as different techniques apart from maximum and algebraic sum aggregation. Additionally, considering the use of a greater number of partitions for each fuzzy control variable can be proposed to enhance the precision and accuracy of the findings.

REFERENCES

[1] Manoj Kr. Dutta, "Performance Analysis of Deflection Routing and Segmentation Dropping Scheme in Optical Burst Switching (OBS) Network: A Simulation Study," Springer Nature Singapore, Advances in Intelligent Systems and Computing 988 Vol. 988, Proceedings of Second International Conference on Computational Intelligence, 121-128, 2020, doi:10.1007/978-981-13-8222-2_10.

[2] Kavitha GR and Indumathi T.S, IJECE," Novel ROADM modelling with WSS and OBS to Improve Routing Performance in Optical Network," International Journal of Electrical and Computer Engineering, Vol. 6, No. 2, April 2016, pp. 666~673 ISSN: 2088-8708, DOI: 10.11591/ijece.v6i2.8300

[3] Hani A. M. Harb, Waleed M. Gaballah, Ahmed S. Samra and Arief Marwanto, "A Study of the Number of Wavelengths Impact in the Optical Burst Switching Core Node," IEEE, Proc. EECSI 2017, Yogyakarta, Indonesia,19-21 September 2017.

[4] Rabia Tahir Bajwa, 2018, "A Comparative Study of Optical Burst Switching Network Techniques", INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY SCIENCES AND ENGINEERING, VOL. 9, NO. 5, MAY 2018, [ISSN: 2045-7057] www.ijmse.org

[5] V.Kishen Ajay Kumar, K. Suresh Reddy and M. N. Giri Prasad, Springer, "Review of contemporary literature on burst assembling and routing strategies in OBS networks,"J Opt (September 2018) 47(3):324–331, 9 March 2018, doi:10.1007/s12596-018-0454-1.

[6] P. Boobalan, M. Mathimanirangan, V. V. Kumar, and M. Barakathulla, "Hybrid Optical Switching In Optical Code Division Multiplexing Networks," International Journal ofResearch in Engineering and Technology, vol. 3, no. 2, pp. 663-667, 2014

[7] Laila A. Wahab Abdullah Naji et al, March 2022, "Performance analysis comparison of optical burst switching networks" contention resolution

techniques", Indonesian Journal of Electrical Engineering and Computer Science Vol. 25, No. 3, pp. 1539~1548 ISSN: 2502-4752, DOI: 10.11591/ijeecs.v25.i3.pp 1539-1548

[8] Li Shuo, February 2014, "Analysis and Synthesis of Optical Burst Switched Networks", Department of Electronic Engineering, for the degree of Doctor of Philosophy City University Of Hong Kong, Thessis.

[9] Abdulsalam Abdullah Mohammed Yayah, 2012, "Improving End-To-End Delay Of Optical-Burst-Switching Networks Through Enhanced Burst-Assembly And Offsettime Scheme", thesis, Master of Science (Computer Science), page

[10] Richa Awasthi, Lokesh Singh, Asifullah Khan, 2017, "Estimation Of Bursts length And Design Of A Fiber Delay Line Based OBS Router ", Journal of Engineering Science and Technology Vol. 12, No. 3.

[11] Ibrahim Khider,Laila A. Wahab , Hadeil Haydar, 2023**), "**Impact of Fuzzy Offset Time on Delay and Burst Loss Ratio for Optical Burst Switching Networks", International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249-8958,Volume-12, Issue-3, February 2023

[12] Mirosław Klinkowski et al, 2009, "Performance Overview of the Offset Time Emulated OBS Network" Journal Of Lightwave Technology, Vol. 27, No. 14, July 15, 2009, 0733-8724/$25.00 © IEEE.

[13] Li, J., Qiao, C., Xu, J., Xu, D., 2007. Maximizing Throughput for Optical Burst Switching Networks. IEEE/ACM Transactions on Networking 15 (5): 1163-1176.

[14] T. Venkatesh C. Siva Ram Murthy, "An analytical Approach to Optical Burst Switched Networks, ISBN 978-1-4419-1509-2, e-ISBN 978-1-4419-1510-8, DOI 10.1007/978-1-4419-1510-8, Springer, New York Dordrecht Heidelberg London Library of Congress Control Number: 2009939338, c Springer Science+Business Media, LLC 2010, Book

[15] Hazem (Moh'd Said), Abdel Majid Hatamleh November 2016 "Contention and Scheduling Algorithms in Optical Burst Switched Networks" (152279 – 0764).

[16] B. Nleya and A. Mutsvangwa, "Enhanced Congestion Management forMinimizing Network Performance Degradation In OBS Networks", Department of Electronic Engineering, Steve Biko Campus, Durban University of Technology, Durban 4001, South Africa. E-mail: bmnleya@ieee.org, Faculty of Education, Mafikeng Campus, North West University, Mmabatho, 2735, South Africa, 2018, **Page(s):** 48 – 57 Volume: 109, Issue: 1, March 2018)

[17] Barpanda, R. S., Turuk, A. K., & Sahoo, B. (2018). QoS aware routing and wavelength allocation in optical burst switching networks using diferential evolution optimization. Digital Communications and Networks, 4(1), 3–12

[18] Reza Poorzare et. al, Vol. 5 No.1 19 March 2018 "Improving optical burst switching networks (OBS) performance by adjusting maximum burst size and burstification time" International information and Engineering technology Association (IIETA). Review of Computer Engineering Studies.

[19] Reza Poorzare and Siamak Abedidarabad, November 14, 2018, "A Novel Implementation of TCP Vegas by Using A Fuzzy-Threshold Base Algorithm to Improve Performance of Optical Networks

[20] Reza Poorzare and Siamak Abedidarabad, July 25, 2019 "A Brief Review on the Methods that Improve Optical Burst Switching Network Performance", J. Opt. Commun. 2019; aop.

[21] Abdulsalam A. Yayah1 et.al, 2019, Springer, "Intelligent Offset Time Algorithm for Optical Burst Switching Networks", © Springer Nature Switzerland AG 2019 F. IRICT 2018, AISC 843, pp. 427–439, 2019. doi.: 10.1007/978-3-319-99007-1_41

Journal of Intelligent System and Applied Data Science (JISADS)

Journal homepage : https://www.jisads.com

ISSN (2974-9840) Online

Machine Learning Techniques for Resource Management: A Survey Study

Hadiel Elshaik[1]*, *Salahe*ldin Edam[1]

hdola1989rm@gmail.com

[1]School of Electronic Engineering, Sudan University of Science and Technology, Khartoum, Sudan

Abstract

The study's objective was to use machine learning techniques to provide an overview of resource management issues. In order to demonstrate how resource management machine learning algorithm's function, the study uses a qualitative methodology. The results showed that the efficient deployment of very heterogeneous IoT networks depends on resource management, among other aspects of network administration. In conclusion, IoT networks struggle with resource allocation in addition to having to decide how to manage resources based on various contexts and situations. ML and DL models, unlike traditional resource management techniques such as optimization and heuristics-based methods, game theoretical and cooperative approaches, can derive actions from run-time context information and can retune and re-train themselves in response to changes in the environment. ML and DL approaches offer significant potential for managing and making decisions in IoT applications that are large-scale, complex, distributed, and dynamic. These approaches are particularly promising in addressing challenges related to model uncertainty, interpretability, training costs, and generalization from test workloads to real-world user workloads. To effectively tackle radio resource management issues in the expanding IoT networks, it is crucial to carefully design solutions and conduct further scientific research in the future.

Keywords*:* Machine Learning (ML); Resource Management; virtual machines (VMs); Deep Learning (DL); Reinforcement Learning (RL); Artificial Intelligence (AI); Heterogeneous Networks (HetNets)

1. INTRODUCTION

A range of resource management functions are handled by machine learning, including workload estimation, task scheduling, VM consolidation, resource optimization, and energy optimization (Khan et al., 2022). Computing's emergence as a fifth utility is presently underway possible as a result of the environment that cloud computing has created for consumers of software and IT infrastructure (Buyya et al., 2018). In cloud computing, data center resource management remains a tough problem that is significantly influenced by application workload. Traditional cloud computing infrastructures, such as data centers, where applications were connected to individual physical servers, were frequently over-provisioned in order to manage issues with the highest workload (Xu et al., 2017). Because of the waste of resources and floor space, the data center was costly to run in terms of resource management. On the other hand, virtualization technology has demonstrated its ability to make data centers easier to administer. Among the several benefits of this technology are server consolidation and increased server utilization. Large-scale technology giants like Amazon, Google, and Microsoft operate extensive

data centres that require sophisticated resource management. These data centres encompass various components, including servers, virtual machines (VMs), and other associated management responsibilities (Bianchini et al., 2020). Many VMs with varying workload types and quantities are assigned to a server host in these data centers. Because of the variable and irregular demand, a server may be over- and underutilized, resulting in an imbalance in the resource use assigned to virtual machines on a certain hosting server. This could lead to issues such as inconsistent quality of service (QoS), unbalanced energy use, and SLA violations (Singh and Kumar, 2019).

In an uneven workload scenario, the average CPU and memory utilization was found to be 17.76% and 77.93%, respectively. However, studies conducted in Google data centres have shown that the CPU utilization in a Google cluster does not exceed 60%, while the memory utilization remains below 50% (Kumar et al., 2021). Workload inconsistency diminishes data center productivity, which has an impact on energy consumption. It is proportionate to the data centre's financial loss and operational expenses. Excess energy consumption impacts carbon footprints directly, and we must look for alternative and eliminate it because an ideal machine absorbs more than half of maximum energy usage. (Barroso et al., 2013). According to an EIA (Energy Information Administration) survey, data centres consumed around 35 Twh (Tera Watt hour) of energy in 2015 and will consume 95 Twh by 2040 (Khan et al., 2022). Determining the optimal mapping of virtual machines (VMs) to servers is crucial for balancing resource utilization and reducing the number of active servers (Li et al., 2013). However, this problem is challenging and falls under the category of NP-complete. To ensure quality of service (QoS) standards and maximize the benefits of data centres, it is essential to have an effective resource management strategy (Kumar and Singh, 2020). Intelligent mechanisms in the future will provide insights that enable applications to map to machines with higher resource utilization (Kumar et al., 2020). Predicting these future insights is challenging due to the nonlinear and dynamic nature of VM workloads.

Nevertheless, there are two methods for obtaining future workload insights: historical workload-based prediction methods, which learn trends from historical workload data, and homeostatic-based prediction methods, which estimate future workload by subtracting the prior workload from the current workload (Kumar and Singh, 2018). The mean of the prior workload can be either static or dynamic. Both approaches have advantages and disadvantages, but historical forecasts are considered more straightforward and well-established in this field (Khan et al., 2022).

Intelligent resource management will be crucial in maximizing the data center's SLA, energy consumption, and operational expenses by performing efficient and intelligent resource provisioning. Data center resource management includes tasks such as resource provisioning, reporting, workload scheduling, and a range of other responsibilities (Ilager et al., 2020). The provisioning of resources is central to many of these procedures. The purpose of resource provisioning is to provide cloud resources to virtual machines (VMs) in response to end-user requests while limiting SLA violations related to availability, dependability, response time constraints, and cost limits (Shahidinejad et al., 2021).

To minimize over- or under-provisioning, it should assign resources based on end-user requirements, such as allocating more or less resources to VMs. This resource allocation approach can be used in two ways: proactive and reactive.

Proactive tactics focus on anticipating future workloads by using historical workload trends as a guide, whereas reactive operations are carried out when resource demand emerges. As a result, It may be concluded that the experience of historical-based prediction methods can be successfully merged into proactive methods to provide intelligent dynamic resource scaling, hence promoting intelligent dynamic resource management. Furthermore, based on projections, different operations such as task scheduling, thermal management, and VM consolidation may be carried out to improve QoS and optimize resource utilization and energy usage. Machine learning (ML) techniques are used in a variety of industries, including computer vision, pattern recognition, and bioinformatics. The advancement of machine learning techniques has benefited large-scale computing systems (Mao et al., 2019). In a recent report, Google outlined its initiatives to optimize electricity use, reduce costs, and boost productivity (Jeff, 2018). The Structure of the research paper shown in Fig 1.

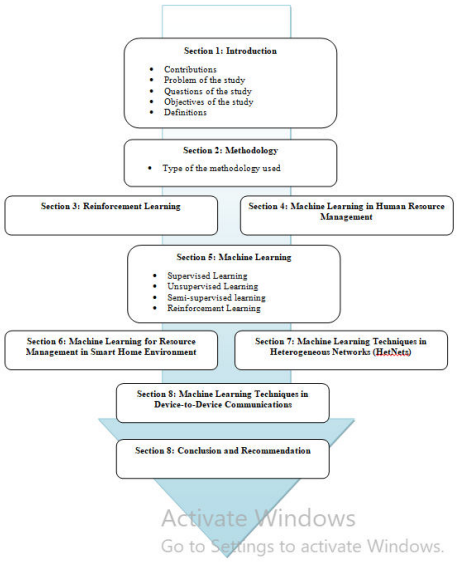

Figure1: Structure of the research paper.

2. REVIEW OF MACHINE LEARNING

Machine learning is used in resource management in a variety of ways, including workload estimation, job scheduling, VM consolidation, resource optimization, and energy optimization (Khan et al., 2022). The application burden has a substantial impact on data center resource management, which is still a complex problem. In conventional cloud computing environments such as data centers, applications were often connected to specific physical servers, and these servers were frequently over-provisioned to manage difficulties with the highest workload (Xu et al., 2017). Major IT behemoths like Google, Microsoft, and Amazon have enormous data centres with complex resource management. Servers, virtual machines (VMs), and other administrative duties are part of these massive data centres' resource management (Bianchini et al., 2020). In these data centres, a server host is assigned to a large number of VMs with varying workload types and amounts. The dynamic and fluctuating demand for resources in virtual machines can cause an imbalance in resource allocation on hosting servers, leading to over-utilization and under-utilization. These problems can result in irregular quality service (QoS), imbalanced energy utilization, and violations of service level agreements (SLAs) (Singh and Kumar, 2019). This paper aims to review the challenges associated with resource management and explores the application of machine learning techniques to address these issues.

Definitions

Machine Learning (ML) can be defined as the capacity to extrapolate knowledge from data and then apply that knowledge to modify the behavior of an ML agent in accordance with the learned information. Techniques for machine learning have been applied to classification, regression, and density estimation applications. IoT devices provide enormous amounts of data, which can be used by data-driven ML approaches to create automated IoT

service solutions. Deep Learning (DL), more particularly machine learning (ML), can be utilized for feature extraction and practical categorization when there is a large and multidimensional amount of data accessible (Hussain et al., 2020).

3. METHODOLOGY

The goal of qualitative research, which was used in this study, is to find notable patterns that are indicative of a specific event through text analysis and interpretation, interviews, and observations (Auerbach and Silverstein, 2003). The steps of ML of wireless sensor networks shown in Fig. 2 [1].

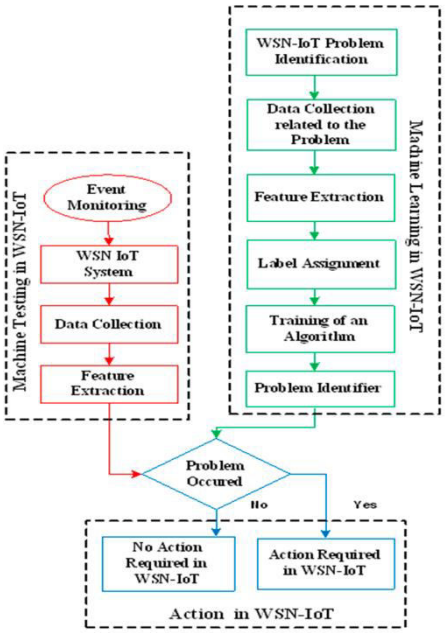

Figure 2: Steps of ML of wireless sensor networks.

Because it focuses on determining the characteristics of the population or subject under investigation, the study employs qualitative research as a research strategy. This qualitative methodology focuses on the "what" rather than the "why" of the study topic. The qualitative research approach focuses on describing the core of a demographic segment rather than "why" a certain occurrence happens. In other words, it "describes" the subject of the inquiry without going into detail about "why" it occurs (Maxwell, 2008).

The study followed the qualitative approach to explain the study (Machine Learning Techniques For Resource Management: A Survey Study) through the literature review.

Reinforcement Learning

We can apply ML approaches when past information about the system, network, users, and parameters is unavailable and needs to be anticipated along with control decisions. Reinforcement Learning (RL) is one of these methods, which involves monitoring system behavior and unknown parameters over time through trial and error in order to determine the best course of action. ML is advised when there is a model or algorithm lack for resource management challenges (Chen et al., 2019).

Model deficit refers to a lack of domain expertise or the absence of mathematical models, whereas algorithm deficit refers to the presence of a well-established mathematical model but difficulty optimizing existing algorithms using it. In this case, lower-complexity ML solutions are desirable. Furthermore, when contextual information is critical to include in the decision-making process, ML techniques are best suited. Because of the enormous number of devices producing massive amounts of data and the unknown system or network states and parameter values, the majority of IoT applications meet the aforementioned characteristics (Hussain et al., 2020).

4. MACHINE LEARNING IN HUMAN RESOURCE MANAGEMENT

Machine learning models are actively progressing in a variety of human resource management roles (Scholz, 2017). Currently, machine learning models are progressing in a

number of areas related to human resource management. This study gives a summary of the important HR functions that can be enhanced by the implementation of machine learning and AI-based solutions. In this paper, three unique conceivable and potential scopes of AI solution implementation are examined, with a focus on three different aspects of employee engagement, organizational culture management, and the appraisal system. Using the decision tree model and logistic regression models to train datasets for an application might enhance the likelihood that the answers will be more accurate and will produce the best form of the evaluation system. If solutions are developed along the lines of what has been discussed, they may be helpful to organizations in managing their strategic human resource practices (Rudra Kumar, 2022).

Machine learning is the study of teaching computers to recognize objects or make predictions without being specifically programmed to do so (Jordan, 2015). Its primary concept is that by using statistical methods and training data, it is feasible to create algorithms that can forecast potential, unforeseen values. Over the previous two decades, machine learning has advanced from a research project to a widely utilized commercial tool. Machine learning has emerged as the go-to technique for creating useful applications in computer vision (Janai et al., 2020), speech recognition (Deng and Li, 2013), natural language processing (Olsson, 2009), robot control (Chin et al., 2020), self-driving cars (Stilgoe, 2018), efficient web search (Bhatia and Kumar, 2008), purchase recommendations (Hastie et al., 2009), and other artificial intelligence fields as shown in Fig. 3.[2].

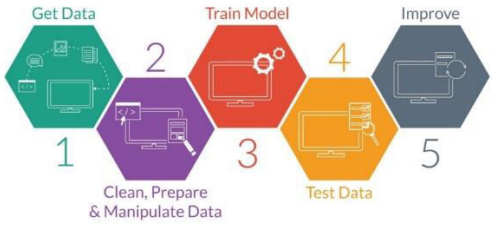

Figure 3: Machine Learning Process

Many AI system developers now understand that training a system by providing examples of acceptable input-output activities is significantly easier than manually programming it by making predictions for all possible inputs for various purposes. This accomplishment is primarily due to the availability of massive amounts of data as well as better server and GPU processing power efficiency [Goodfellow rt al., 2016]. Depending on the modeling aim and the issue at hand (RL), machine-learning algorithms are classified as supervised learning, semi-supervised learning (SSL), unsupervised learning, and reinforcement learning. Unsupervised learning is divided into two categories: clustering and dimension reduction (Hartigan et al., 1979; Guha et al., 2000; Ding et al., 2002), whereas supervised learning, is divided into two categories: regression problem and classification problem (e.g., sentence classification (Yoon, 2014; Wenpeng and Schütze, 2015), picture classification (Yang et al, 2009; Bazi and Melgani, 2009; Ciregan et al., 2013), etc.

Supervised Learning (Sen et al., 2020): In supervised learning, each data sample consists of a name and multiple input attributes. The objective of the learning process is to create a mapping function that accurately relates the input features to the corresponding label. This mapping function can then be used to predict the label for new data by utilizing additional input features. Supervised learning is a widely used machine learning approach across various applications. An example of supervised

learning is classification, where an object is assigned to a specific category based on its characteristics, such as classifying a mobile device based on its brand and features. On the other hand, if the objective is to predict a continuous variable, such as stock prices, the supervised learning task is called regression.

Unsupervised Learning (Celebi and Aydin, 2016): When we have input features without corresponding labels, we are involved in unsupervised learning instead of supervised learning. Unsupervised learning aims to understand the underlying data distribution and explore patterns or differences among data points. A well-known example of unsupervised learning is the clustering problem, which aims to identify meaningful groups within the data, such as grouping virtual machines based on resource utilization patterns. Figure 4 [2] illustrates different machine learning methods, including unsupervised learning.

Semi-supervised learning [Van Engelen and Hoos, 2020]: This subfield of machine learning aims to combine these two tasks, leveraging information from one task to improve the performance of the other. Semi-supervised learning (SSL) algorithms often utilize unlabelled data points to enhance the classification process, for example, by utilizing additional data points with unknown labels. On the other hand, understanding the similarity or belongingness of certain data points to the same class can assist in guiding the clustering process. By incorporating labelled and unlabelled data, SSL approaches can benefit from the advantages of supervised and unsupervised learning to improve overall learning and inference capabilities.

Kober et al. (2013) define Reinforcement Learning as: RL differs from both supervised and unsupervised learning in several aspects. Unlike supervised learning, RL does not require labelled input/output pairs or explicit correction of inferior choices during training. Instead, RL involves an agent interacting with an environment, learning to make decisions through a balance between exploration and exploitation. The agent receives feedback in the form of rewards or penalties based on its actions, which guide its learning process to optimize long-term cumulative rewards. RL is particularly suitable for problems where an agent learns through trial and error to achieve a specific goal in dynamic and uncertain environments. The translator pays the agent for making wise choices or acting in a certain way. If not, it would be approved. Robotics and computer game agent science frequently employ reinforcement learning.

Figure 4: Types of machine learning

5. MACHINE LEARNING FOR RESOURCE MANAGEMENT IN SMART HOME ENVIRONMENT

Smart home applications are highly popular in the realm of IoT, as they integrate various technologies such as security cameras, handheld scanners, tablets, smart appliances, and wireless sensors. These devices often have diverse access and quality-of-service (QoS) requirements, and they access network resources in a random manner. To address resource allocation and random-access challenges within the smart home environment, ML methods like Q-learning and multi-armed bandit can prove beneficial. These techniques enable effective management and optimization of resources in smart

homes. This is so that various methods of reinforcement learning may change with the network environment and learn to do so dynamically (Ali et al., 2020; Kanoun et al., 2016.).

Additionally, small sensors and small payload data can be grouped using K-means clustering and PCA, respectively. Be aware that running conventional optimization and heuristics-based techniques on small, low-cost, and energy-constrained sensors can be quite expensive computationally. Furthermore, because of the heterogeneity of the devices and the overhead associated with information updates and exchange, traditional game theoretical approaches might not be appropriate (Hussain, 2020).

6. MACHINE LEARNING TECHNIQUES IN HETEROGENEOUS NETWORKS (HETNETS)

Resource management in HetNets, where cellular and small cell users coexist with different Radio Access Technologies (RATs), poses several challenges, such as cross- and co-tier interference, mobility management, user association, RAT selection, and self-organization. To address these difficulties, researchers have explored ML-based methods alongside traditional techniques like optimization and heuristics (Omar et al., 2017). For instance, Simsek et al. (2015) developed an RL-based mechanism for inter-cell coordination and handover in HetNets, enabling devices to learn effective resource management. Vasudeva et al. (2017) utilized fuzzy game-theory techniques to reduce network energy consumption while maintaining QoS levels. Perez et al. (2017) proposed a unique cognitive RAT selection paradigm using ML methods.

Furthermore, ML approaches have also been employed for network self-organization, encompassing self-configuration, self-organization, and self-healing. Fan and Sengul (2014) investigated the use of artificial neural networks (ANN) for self-optimization in HetNets, while Alqerm and Shihada (2018) proposed an efficient resource allocation system utilizing online learning algorithms and Q-value theory for QoS provisioning at high data rates. A similar study on resource distribution in heterogeneous cognitive radio networks can be found in Fan and Sengul (2014).

7. MACHINE LEARNING TECHNIQUES IN DEVICE-TO-DEVICE COMMUNICATIONS

D2D networks allow two devices in close proximity to connect to one another without the need for a centralized base station. A D2D network offloads traffic from the primary BS by utilizing proximity communications, boosting the network's spectrum efficiency and Energy Efficiency (EE) (Ansari et al., 2017). Low route loss allows for high spectrum efficiency and sum rate, while low transmission power between radios guarantees EE. Numerous D2D network-related topics, including resource and power allocation, mode selection, proximity sensing, and interference avoidance, have been treated in the literature. (Ahmed et al., 2018; Liu et al., 2019). Recently, machine learning (ML) has been used to handle a range of D2D communication difficulties, including caching (Cheng et al., 2018), security and privacy [Haus et al., 2017], and others.

The efficient use of scarce resources to meet the QoS requirements of all network entities, including cellular and D2D users, presents a significant problem for D2D networks. The study of Maghsudi and Stańczak, (2014) developed a bandit-based channel access method for a distributed D2D system in which each pair chooses the best channel for communication. This raises the rates of individual D2D pairs while simultaneously reducing interference from other users sharing the same

channels. Similarly to that, Asheralieva and Miyanaga, (2016) presented another study on channel selection with autonomous learning. The best resource selection technique is identified using Q-learning in a method for resource allocation in D2D networks that was published by Luo et al., (2014). Similar to Khan et al., (2017), the authors used cooperative RL to increase the individual device throughputs and the sum rates of the system by employing cooperative strategy planning to allocate resources optimally. In order to create an energy-efficient network solution, ML was used to optimize power allocations for various D2D couples (AlQerm and Shihada, 2017).

8. CONCLUSION

The aim of the study is to explain a review of problems with resource management utilizing machine learning techniques. The study employs a qualitative methodology to illustrate how resource management machine learning algorithms work. We employ the qualitative method to identify notable patterns that are suggestive of a specific occurrence.

In the context of IoT networks, resource management plays a vital role alongside other network administration tasks. The successful implementation of diverse IoT networks requires effective resource allocation and contextually appropriate resource management decisions. Unlike traditional approaches based on optimization and heuristics, game theoretical and cooperative methods are being utilized. ML and DL models, on the other hand, have the capability to adapt and retrain themselves by inferring actions from real-time context information in response to environmental changes. Particularly in complex, large-scale, distributed, and dynamic IoT application scenarios, ML and DL approaches hold significant promise for automating resource management and decision-making processes.

To solve complex radio resource management problems in emerging IoT networks, we recommend future scientific research. We must carefully build solutions for these networks in response to issues such as model uncertainty, model interpretability, model training costs, and generalization from test workloads to real application user workloads.

9. REFERENCES

[1] Asheralieva, A., & Miyanaga, Y. (2016). An autonomous learning-based algorithm for joint channel and power level selection by D2D pairs in heterogeneous cellular networks. IEEE transactions on communications, 64(9), 3996-4012.

[2] Cheng, P., Ma, C., Ding, M., Hu, Y., Lin, Z., Li, Y., & Vucetic, B. (2018). Localized small cell caching: A machine learning approach based on rating data. IEEE Transactions on Communications, 67(2), 1663-1676.

[3] Fang, Z. (2010). Resource management on cloud systems with machine learning (Master's thesis, Universitat Politècnica de Catalunya).

[4] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[5] Ali, S., Ferdowsi, A., Saad, W., Rajatheva, N., & Haapola, J. (2020). Sleeping multi-armed bandit learning for fast uplink grant allocation in machine type communications. IEEE Transactions on Communications, 68(8), 5072-5086.

[6] Ding, C., He, X., Zha, H., & Simon, H. D. (2002, December). Adaptive dimension reduction for clustering high dimensional data. In 2002 IEEE International Conference on Data Mining, 2002. Proceedings. (pp. 147-154). IEEE.

[7] Auerbach, C., & Silverstein, L. B. (2003). Qualitative data: An introduction to coding and analysis (Vol. 21). NYU press.

[8] Bhatia, M. P. S., & Kumar, A. (2008). Information retrieval and machine learning: supporting technologies for web mining

research and practice. Webology, 5(2), 5.

[9] Bazi, Y., & Melgani, F. (2009). Gaussian process approach to remote sensing image classification. IEEE transactions on geoscience and remote sensing, 48(1), 186-197.

[10] Bianchini, R., Fontoura, M., Cortez, E., Bonde, A., Muzio, A., Constantin, A. M., ... & Russinovich, M. (2020). Toward ml-centric cloud platforms. Communications of the ACM, 63(2), 50-59.

[11] Buyya, R., Srirama, S. N., Casale, G., Calheiros, R., Simmhan, Y., Varghese, B., ... & Shen, H. (2018). A manifesto for future generation cloud computing: Research directions for the next decade. ACM computing surveys (CSUR), 51(5), 1-38.

[12] Ciregan, D., Meier, U., & Schmidhuber, J. (2012, June). Multi-column deep neural networks for image classification. In 2012 IEEE conference on computer vision and pattern recognition (pp. 3642-3649). IEEE.

[13] Chen, M., Challita, U., Saad, W., Yin, C., & Debbah, M. (2019). Artificial neural networks-based machine learning for wireless networks: A tutorial. IEEE Communications Surveys & Tutorials, 21(4), 3039-3071.

[14] Ahmed, M., Li, Y., Waqas, M., Sheraz, M., Jin, D., & Han, Z. (2018). A survey on socially aware device-to-device communications. IEEE Communications Surveys & Tutorials, 20(3), 2169-2197.

[15] Barroso, L. A., Clidaras, J., & Hölzle, U. (2013). The datacenter as a computer: An introduction to the design of warehouse-scale machines. Synthesis lectures on computer architecture, 8(3), 1-154.

[16] Guha, S., Rastogi, R., & Shim, K. (2000). ROCK: A robust clustering algorithm for categorical attributes. Information systems, 25(5), 345-366.

[17] Celebi, M. E., & Aydin, K. (Eds.). (2016). Unsupervised learning algorithms. Berlin: Springer International Publishing.

[18] Deng, L., & Li, X. (2013). Machine learning paradigms for speech recognition: An overview. IEEE Transactions on Audio, Speech, and Language Processing, 21(5), 1060-1089.

[19] Ansari, R. I., Chrysostomou, C., Hassan, S. A., Guizani, M., Mumtaz, S., Rodriguez, J., & Rodrigues, J. J. (2017). 5G D2D networks: Techniques, challenges, and future prospects. IEEE Systems Journal, 12(4), 3970-3984.

[20] Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. Journal of the royal statistical society. series c (applied statistics), 28(1), 100-108.

[21] Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). The elements of statistical learning: data mining, inference, and prediction (Vol. 2, pp. 1-758). New York: springer.

[22] Haus, M., Waqas, M., Ding, A. Y., Li, Y., Tarkoma, S., & Ott, J. (2017). Security and privacy in device-to-device (D2D) communication: A review. IEEE Communications Surveys & Tutorials, 19(2), 1054-1079.

[23] Hussain, F., Hassan, S. A., Hussain, R., & Hossain, E. (2020). Machine learning for resource management in cellular and IoT networks: Potentials, current solutions, and open challenges. IEEE communications surveys & tutorials, 22(2), 1251-1275.

[24] Ilager, S., Muralidhar, R., & Buyya, R. (2020, October). Artificial intelligence (ai)-centric management of resources in modern distributed computing systems. In 2020 IEEE Cloud Summit (pp. 1-10). IEEE.

[25] Sen, P. C., Hajra, M., & Ghosh, M. (2020). Supervised classification algorithms in machine learning: A survey and review. In Emerging technology in modeling and graphics (pp. 99-111). Springer, Singapore.

[26] Kumar, J., Singh, A. K., & Buyya, R. (2021). Self-directed learning-based workload forecasting model for cloud resource management. Information Sciences, 543, 345-366.

[27] Stilgoe, J. (2018). Machine learning, social learning and the governance of self-driving cars. Social studies of science, 48(1), 25-56.

[28] Xu, M., Tian, W., & Buyya, R. (2017). A survey on load balancing algorithms for virtual machines placement in cloud computing. Concurrency and Computation: Practice and Experience, 29(12), e4123.

[29] Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260.

[30] Scholz, T. M. (2017). Big data in organizations and the role of human resource management: A complex systems theory-based conceptualization. Frankfurt a. M.: Peter Lang International Academic Publishers.

[31] Kumar, J., & Singh, A. K. (2020). Cloud datacenter workload estimation using error preventive time series forecasting models. Cluster Computing, 23(2), 1363-1379.

[32] Liu, S., Wu, Y., Li, L., Liu, X., & Xu, W. (2019). A two-stage energy-efficient approach for joint power control and channel allocation in D2D communication. IEEE Access, 7, 16940-16951.

[33] Maghsudi, S., & Stańczak, S. (2014). Channel selection for network-assisted D2D communication via no-regret bandit learning with calibrated forecasting. IEEE Transactions on Wireless Communications, 14(3), 1309-1322.

[34] Wenpeng, Y., & Schütze, H. (2015). Multichannel variable-size convolution for sentence classification [C]. In Proc of the 19th Conf on Computational Natural Language Learning. Stroudsburg, PA: ACL (pp. 204-214).

[35] Janai, J., Güney, F., Behl, A., & Geiger, A. (2020). Computer vision for autonomous vehicles: Problems, datasets and state of the art. Foundations and Trends® in Computer Graphics and Vision, 12(1–3), 1-308.

[36] Kumar, J., Singh, A. K., & Buyya, R. (2020). Ensemble learning based predictive framework for virtual machine resource request prediction. Neurocomputing, 397, 20-30.

[37] Yoon, K. (2014). Convolutional Neural Networks for Sentence Classification [OL]. arXiv Preprint.

[38] Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., & Alizadeh, M. (2019). Learning scheduling algorithms for data processing clusters. In Proceedings of the ACM special interest group on data communication (pp. 270-288).

[39] Khan, T., Tian, W., Zhou, G., Ilager, S., Gong, M., & Buyya, R. (2022). Machine learning (ML)–Centric resource management in cloud computing: A review and future directions. Journal of Network and Computer Applications, 103405.

[40] Simsek, M., Bennis, M., & Güvenç, I. (2015, March). Context-aware mobility management in HetNets: A reinforcement learning approach. In 2015 ieee wireless communications and networking conference (wcnc) (pp. 1536-1541). IEEE.

[41] Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. Machine Learning, 109(2), 373-440.

[42] Perez, J. S., Jayaweera, S. K., & Lane, S. (2017, June). Machine learning aided cognitive RAT selection for 5G heterogeneous networks. In 2017 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom) (pp. 1-5). IEEE.

[43] Yang, J., Yu, K., Gong, Y., & Huang, T. (2009, June). Linear spatial pyramid matching using sparse coding for image classification. In 2009 IEEE Conference on computer vision and pattern recognition (pp. 1794-1801). IEEE.

[44] Omar, M. S., Hassan, S. A., Pervaiz, H., Ni, Q., Musavian, L., Mumtaz, S., & Dobre, O. A. (2017). Multiobjective optimization in 5G hybrid networks. IEEE Internet of Things Journal, 5(3), 1588-1597.

[45] Shahidinejad, A., Ghobaei-Arani, M., & Masdari, M. (2021). Resource provisioning using workload clustering in cloud computing environment: a hybrid approach. Cluster Computing, 24(1), 319-342.

[46] Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260.

[47] Li, X., Qian, Z., Lu, S., & Wu, J. (2013). Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. Mathematical and Computer Modelling, 58(5-6), 1222-1235.

[48] Jeff, D. (2018). ML for system, system for ML, keynote talk in Workshop on ML for Systems, NIPS.

[49] Luo, Y., Shi, Z., Zhou, X. I. N., Liu, Q., & Yi,

Q. (2014, December). Dynamic resource allocations based on Q-learning for D2D communication in cellular networks. In 2014 11th international computer conference on wavelet active media technology and information processing (ICCWAMTIP) (pp. 385-388). IEEE.

[50] Kanoun, K., Tekin, C., Atienza, D., & Van Der Schaar, M. (2016). Big-data streaming applications scheduling based on staged multi-armed bandits. IEEE Transactions on Computers, 65(12), 3591-3605.

[51] Kumar, J., & Singh, A. K. (2018). Workload prediction in cloud using artificial neural network and adaptive differential evolution. Future Generation Computer Systems, 81, 41-52.

[52] Simsek, M., Bennis, M., & Güvenç, I. (2015, March). Context-aware mobility management in HetNets: A reinforcement learning approach. In 2015 IEEE Wireless Communications and Networking Conference (WCNC) (pp. 1536-1541). IEEE.

Journal of Intelligent System and Applied Data Science (JISADS)

Journal homepage : https://www.jisads.com

ISSN (2974-9840) Online

Emerging Trends in Cloud Computing: A Comprehensive Analysis of Deployment Models and Service Models for Scalability, Flexibility, and Security Enhancements.

*Rajesh komar1, Arjun Patil 1**
Ragishkaa22@Gmail.Com, patil.Arjun2002@gmail.Com

1 Navodaya Institute of Technology, Raichur.

Abstract

Cloud computing has revolutionized IT infrastructure management and service delivery across industries. This study provides a comprehensive analysis of deployment models and service models in cloud computing, focusing on their significance and implications for organizations. By examining each model's features, benefits, challenges, and intrusion threats, decision-makers can make informed choices and implement adequate security measures. The research contributes to existing knowledge by offering insights into cloud computing and recommendations for secure adoption. The study begins with an overview of cloud computing, highlighting its scalability and flexibility. It then explores deployment models (public, private, hybrid, community) and service models (IaaS, PaaS, SaaS), assessing their characteristics and use cases. Intrusion threats are discussed, emphasizing the need for robust security measures. Real-world case studies showcase successful models and security strategies. This study equips organizations with the knowledge to leverage cloud computing while safeguarding their systems and data.

Keywords*:* Cloud computing, Service models, Cloud management, Cloud threats

1. INTRODUCTION

Cloud computing has emerged as a dominant technology paradigm for managing IT infrastructure and delivering services in various sectors. Accessing computing resources on-demand over the internet has revolutionized how organizations operate, providing scalability, cost-effectiveness, and flexibility. However, with the widespread adoption of cloud computing, new challenges and risks, particularly in the areas of deployment models and service models, have come to the forefront.

This comprehensive study aims to provide an in-depth analysis of deployment and service models in cloud computing, highlighting their significance and implications for organizations. By examining the features, benefits, challenges, and potential intrusion threats associated with each model, this research aims to assist decision-makers in making informed choices and implementing effective security measures.

To establish a strong foundation, it is essential to understand the existing body of knowledge on cloud computing and its various aspects. The work in [1] provides a comprehensive view of cloud computing, highlighting its essential characteristics and advantages. Additionally, the NIST Definition of Cloud Computing by Mell and Grance [2] offers a widely accepted definition and framework for cloud computing, providing a basis for further exploration.

Deployment models play a crucial role in determining the architecture and accessibility of cloud-based systems. The public cloud model, characterized by shared infrastructure and services, is explored in the research conducted by Buyya et al. [3] and Vaquero et al. [4]. On the other hand, private cloud models dedicated to a single organization are discussed extensively in the literature (Dillon et al.

[5]; Rittinghouse & Ransome [6]). The hybrid cloud model, combining public and private cloud elements, and the community cloud model, shared among organizations with common interests, are also examined in this study.

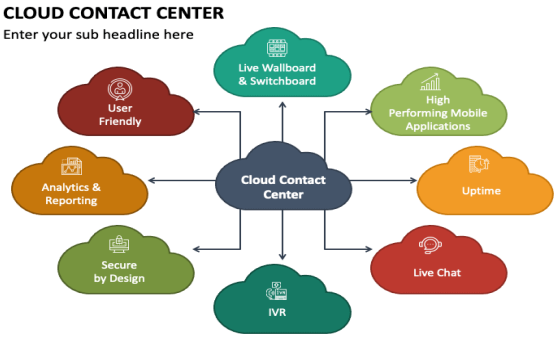

Fig. 1 Cloud contact centre

Service models provide different levels of abstraction and functionalities in cloud computing. The Infrastructure as a Service (IaaS) model, which offers virtualized computing resources, is explored in the research by Subashini and Kavitha [7]. The Platform as a Service (PaaS) model, providing a development and deployment platform, is discussed by Wang et al. [8]. Finally, Software as a Service (SaaS) model enabling access to software applications over the internet, is examined in the works of Hamdaqa et al. [9] and Rimal et al. [10].

While the benefits of cloud computing are evident, security remains a critical concern. Intrusion threats pose risks to cloud-based systems, necessitating a comprehensive understanding of potential vulnerabilities. Ristenpart et al. [11] highlight the need to explore information leakage in third-party compute clouds, shedding light on the intrusion threats associated with cloud computing environments. Furthermore, Liang et al. [12] present a comprehensive study on intrusion detection in the cloud, emphasizing the importance of adequate security measures.

Organizations can make informed decisions and implement robust security strategies by delving into the nuances of deployment and service models in cloud computing and considering the potential intrusion threats. This study contributes to the existing body of knowledge by providing a comprehensive analysis, paving the way for the secure and effective adoption of cloud computing in organizations.

2. BACKGROUND AND LITERATURE REVIEW

2.1 Cloud Computing: An Overview

Cloud computing has emerged as a transformative technology in IT infrastructure management and service delivery. It allows organizations to access and utilize virtualized computing resources over the internet, providing scalability, cost-efficiency, and flexibility [13]. This model has revolutionized businesses by enabling on-demand resource provisioning, dynamic scalability, and reduced infrastructure costs.

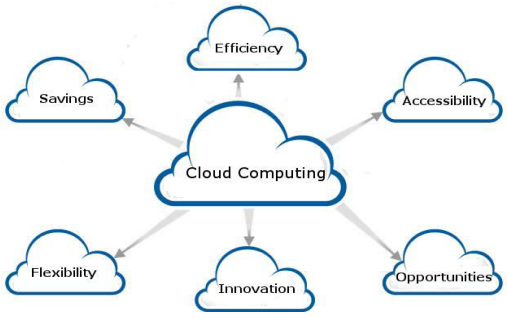

Fig. 2 Cloud specifications

2.2 Deployment Models in Cloud Computing

Deployment models play a crucial role in defining the architecture and ownership of cloud infrastructure. The public cloud model, provided by third-party service providers, offers a shared environment accessible to multiple users (Almorsy et al. [14]). Private clouds, on the other hand, are dedicated to a single organization, providing enhanced control and security (Rimal et al.[15]). Hybrid clouds combine public and private cloud environments, allowing organizations to leverage the benefits of both models (Hassan et al. [16]). Community clouds are shared among organizations with common interests, such as those within the same industry or adhering to specific regulations (Liu et al. [17]).

2.3 Intrusion Threats in Cloud Computing

The security of cloud computing environments is of utmost importance due to potential intrusion threats. Intruders may attempt to exploit vulnerabilities in the system to gain unauthorized access, compromise data confidentiality, or disrupt services. Therefore, it is essential to comprehend and mitigate these intrusion

threats to ensure the integrity and security of cloud-based systems.

Intrusion detection systems are vital in identifying and responding to potential attacks in cloud environments. These systems employ various techniques, including anomaly detection and signature-based methods, to detect and mitigate intrusion attempts. Anomaly detection techniques analyze system behaviour and network traffic patterns to identify deviations from everyday activities, while signature-based methods match known patterns of malicious behaviour to detect specific attacks (Zhang et al.[18]).

Fig. 3 Intrusion Threats in Cloud Computing

Researchers are exploring advanced techniques and approaches to enhance the effectiveness of intrusion detection and response mechanisms in cloud computing. Machine learning algorithms, such as support vector machines (SVMs) and artificial neural networks (ANNs), are being applied to improve the accuracy and efficiency of intrusion detection systems. These algorithms can learn from historical data and adapt to evolving attack patterns, enabling proactive threat detection and timely response.

Furthermore, integrating threat intelligence feeds and real-time monitoring systems is crucial in addressing intrusion threats in cloud computing. Threat intelligence provides valuable information about known vulnerabilities, attack vectors, and malicious activities, allowing organizations to stay updated on emerging threats and proactively implement appropriate security measures (Rauti et al. [19]). Real-time monitoring systems continuously monitor network traffic, system logs, and user activities to detect and respond to suspicious activities promptly.

By combining advanced intrusion detection algorithms, threat intelligence feeds, and real-time monitoring systems, organizations can strengthen their defences against intrusion threats in cloud computing environments. These approaches enable proactive identification and mitigation of attacks, minimizing the risk of data breaches, service disruptions, and unauthorized access.3.4 Summary of Literature

The existing literature provides valuable insights into various aspects of cloud computing, including deployment models, service models, and intrusion threats. Zheng et al. [20] offer a comprehensive survey on cloud computing security, addressing challenges and mitigation strategies. Teng et al. [21] provide an in-depth exploration of cloud deployment models, highlighting their characteristics and considerations. Finally, Almorsy et al. [22] present a comprehensive perspective on service models in cloud computing, discussing their features and applications.

3. METHODOLOGY

This section explains the research approach, data collection methods, analysis techniques, and criteria for selecting relevant research articles, papers, and industry reports for the comprehensive study on cloud computing deployment and service models.

3.1 Research Approach

For this study, a systematic literature review approach was employed. This approach involves an organized and structured evaluation of existing literature to understand the topic comprehensively. The literature review follows a predefined protocol, ensuring a rigorous and unbiased analysis of the available literature. By adopting this approach, we aimed to capture various perspectives, theories, and findings related to deployment and service models in cloud computing.

3.2 Data Collection Methods

The data collection process involved searching and accessing various academic databases, including IEEE Xplore, ACM Digital Library, and Google Scholar. These databases were selected for their comprehensive computer science and information technology literature coverage. Relevant keywords, such as "cloud computing," "deployment models," "service models," and "intrusion threats," were used to search. The search was performed across title, abstract, and full-text fields to ensure the inclusion of relevant articles.

The inclusion criteria for selecting research articles, papers, and industry reports were based on several factors. First, relevance to the research topic was considered, focusing on publications discussing deployment and service models in cloud computing. Second, with a preference for recent publications, the publication date was supposed to ensure the inclusion of the most up-to-date information. Third, priority was given to peer-reviewed journal articles, conference papers, and reports from reputable sources to ensure the credibility and academic rigour of the selected sources.

3.3 Analysis Techniques

The analysis of the collected literature involved a thorough review and extraction of critical information. The selected articles and reports were carefully read, and relevant data points were extracted, including definitions, characteristics, advantages, and limitations of different cloud computing deployment and service models. The extracted information was then organized and synthesized to identify common themes, patterns, and trends across the literature.

A systematic approach was employed to ensure the reliability and validity of the analysis. The extracted data were cross-checked and reviewed by multiple researchers involved in the study. Any discrepancies or differences in interpretation were resolved through discussion and consensus. This collaborative approach helped minimize bias and ensure the accuracy of the analysis.

3.4 Criteria for Selecting Relevant Research Articles, Papers, and Industry Reports

The criteria used to select relevant research articles, papers, and industry reports were designed to ensure the inclusion of high-quality and reputable sources. The publication date was considered to include recent publications that reflect the latest developments in cloud computing. Relevance to the research topic was a crucial criterion, focusing on publications that specifically addressed deployment and service models in cloud computing.

To ensure academic rigour, priority was given to peer-reviewed journal articles and conference papers. These sources undergo a rigorous review process by experts in the field, ensuring the quality and validity of the research findings. Additionally, reports and publications from reputable industry sources were included to capture practical insights and real-world experiences related to cloud computing deployment and service models.

By employing these rigorous methodologies, we aimed to ensure a comprehensive, objective, and reliable analysis of cloud computing deployment and service models, drawing insights from various scholarly and industry sources.

4. DEPLOYMENT MODELS IN CLOUD COMPUTING

Cloud computing offers different deployment models for organizations based on their requirements and desired resource-sharing levels. The primary deployment models in cloud computing include public, private, hybrid, and community clouds.

Public Cloud: The public cloud deployment model is provided by third-party service providers and offers computing resources over the internet. This model shares resources among multiple organizations, resulting in cost savings and scalability.

Private Cloud: The private cloud deployment model is dedicated to a single organization and offers enhanced control, security, and privacy compared to the public cloud. It is either hosted on-premises or by a third-party provider.

Hybrid Cloud: The hybrid cloud deployment model combines the features of public and private clouds, offering a mix of on-premises infrastructure and off-premises resources. It provides flexibility and agility by allowing organizations to leverage the benefits of both models.

Community Cloud: The community cloud deployment model is shared among organizations with common interests, such as those within the same industry or adhering to specific regulations. It enables resource sharing while maintaining control and security.

Organizations need to evaluate their requirements, data sensitivity, regulatory compliance, scalability needs, and budget to select the appropriate deployment model for their cloud computing environment.

5. SERVICE MODELS IN CLOUD COMPUTING

Cloud computing offers different service models that define organizations' control and responsibility over

their computing resources. The three main service models in cloud computing are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

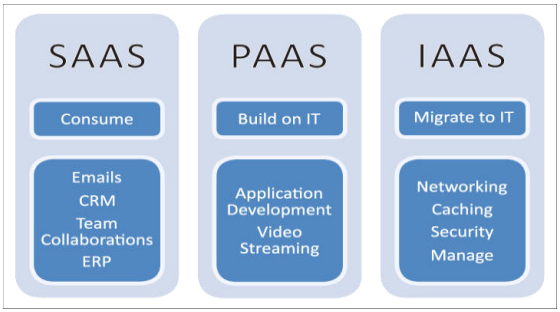

Fig. 4 Service Models in Cloud Computing

5.1 Infrastructure as a Service (IaaS):

IaaS provides virtualized computing resources, including virtual machines, storage, and networks, as a service. Organizations have complete control over the operating systems, applications, and data hosted on the infrastructure. IaaS allows organizations to scale their infrastructure up or down based on demand, providing flexibility and cost-efficiency. It is suitable for organizations that require high control and customization over their computing resources.

5.2 Platform as a Service (PaaS):

PaaS offers a platform for organizations to develop, deploy, and manage applications without controlling the underlying infrastructure. It provides a pre-configured environment that includes the operating system, development tools, and runtime frameworks. PaaS enables organizations to focus on application development and deployment without worrying about infrastructure management. It offers scalability, automatic resource provisioning, and support for multiple programming languages and frameworks.

5.3 Software as a Service (SaaS):

SaaS provides ready-to-use applications and Software over the internet. Organizations access these applications through a web browser or API without the need for installation or maintenance. SaaS offers a range of applications, such as customer relationship management (CRM), enterprise resource planning (ERP), and collaboration tools. It eliminates the need for organizations to manage infrastructure, updates, and maintenance, allowing them to focus on using the Software for their business operations.

6. BENEFITS AND CONSIDERATIONS OF DEPLOYMENT AND SERVICE MODELS

6.1 Benefits of Deployment Models:

Public Cloud: Cost savings, scalability, and accessibility.

Private Cloud: Enhanced control, security, and privacy.

Hybrid Cloud: Flexibility, scalability, and optimized resource allocation.

Community Cloud: Resource sharing, collaboration, and industry-specific solutions.

6.2 Benefits of Service Models:

IaaS: Control, flexibility, and scalability of infrastructure resources.

PaaS: Streamlined application development, automatic resource provisioning, and multi-language support.

SaaS: Easy accessibility, reduced IT management burden, and rapid deployment.

Organizations must consider several factors when choosing the appropriate deployment and service models for their cloud computing environment. These factors include data security and privacy requirements, compliance regulations, scalability needs, cost considerations, and the level of control and customization required.

7. CHALLENGES AND CONSIDERATIONS IN CLOUD DEPLOYMENT

While cloud computing offers numerous benefits, organizations must address several challenges and considerations when deploying cloud-based solutions. Understanding and mitigating these challenges is essential for successful cloud implementation.

7.1 Security and Privacy:

Security and privacy are major concerns in cloud computing. Organizations must protect their data and applications from unauthorized access, breaches, and other security threats. They should employ robust authentication mechanisms, encryption techniques,

and access controls to safeguard sensitive information. Additionally, organizations must understand the cloud service provider's data privacy policies and regulations to ensure compliance with applicable laws and protect user privacy.

7.2 Compliance and Legal Issues:

Organizations must adhere to Certain industries' and regions' specific compliance requirements and regulations when deploying cloud solutions. Compliance standards such as GDPR (General Data Protection Regulation) and HIPAA (Health Insurance Portability and Accountability Act) impose strict guidelines for handling sensitive data. Organizations need to assess the compliance capabilities of their cloud service providers and ensure that their cloud deployment meets the necessary legal and regulatory requirements.

7.3 Data Portability and Vendor Lock-In:

Organizations should consider the ease of migrating their data and applications between different cloud providers or back to an on-premises environment. Vendor lock-in, where organizations become highly dependent on a specific cloud provider's service, can hinder portability and limit flexibility. Evaluating interoperability standards, data formats, and exit strategies upfront can help mitigate the risks of vendor lock-in and ensure data portability.

8. EMERGING TECHNOLOGIES IN CLOUD COMPUTING

Cloud computing continues to evolve, driven by technological advancements and emerging trends. Understanding these trends can provide insights into the future of cloud computing and help organizations make informed decisions about their cloud deployments.

8.1 Edge Computing:

Edge computing aims to bring computing resources closer to the data source or end-users, reducing latency and improving performance. By decentralizing computing power, edge computing enables real-time data processing and analysis, making it ideal for applications that require low latency, such as Internet of Things (IoT) devices. Organizations can leverage edge computing with cloud computing to enhance their overall infrastructure and deliver faster and more responsive services.

8.2 Serverless Computing:

Serverless computing, or Function as a Service (FaaS), allows developers to execute code without explicitly managing or provisioning servers. With serverless computing, organizations pay only for the actual code execution time, leading to cost savings and greater scalability. Serverless architectures simplify application development and deployment, as developers can focus solely on writing code rather than managing infrastructure.

8.3 Multi-cloud and Hybrid Cloud Strategies:

Organizations are increasingly adopting multi-cloud and hybrid cloud strategies to leverage the benefits of multiple cloud providers and combine on-premises and off-premises resources. Multi-cloud environments provide organizations with flexibility, cost optimization, and risk mitigation by distributing workloads across different cloud platforms. Hybrid cloud strategies offer the ability to combine the benefits of private and public clouds, allowing organizations to maintain control over critical data while taking advantage of the scalability and cost-effectiveness of the public cloud.

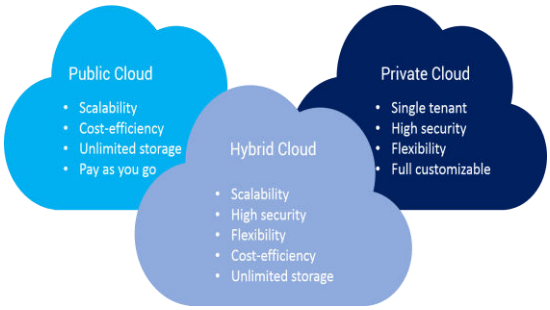

Fig. 5 Hybrid Cloud

9. FUTURE TRENDS AND RESEARCH DIRECTIONS

Cloud computing is a dynamic field that continues to evolve, driven by technological advancements and emerging trends. Several areas of future research and development hold the potential to shape the future of cloud computing.

9.1 Artificial Intelligence and Machine Learning in Cloud Computing:

Integrating artificial intelligence (AI) and machine learning (ML) capabilities into cloud computing can unlock new possibilities for intelligent data analysis, automation, and decision-making. Future research

should focus on developing AI-driven cloud services, optimizing resource allocation for ML workloads, and addressing the challenges of training and deploying ML models in distributed cloud environments.

9.2 Quantum Computing and Cloud Services:

Quantum computing has the potential to revolutionize cloud computing by enabling complex computations and solving problems that are currently infeasible with classical computing. Research efforts should explore the integration of quantum computing with cloud services, such as developing quantum algorithms, enhancing security through quantum encryption, and investigating the scalability and performance of quantum cloud platforms.

9.3 Security and Privacy Enhancements:

As the importance of data security and privacy increases, future research should focus on developing robust security mechanisms and privacy-preserving techniques for cloud computing. Areas of interest include secure data sharing, homomorphic encryption, secure multiparty computation, and advanced threat detection and mitigation strategies to address evolving cybersecurity threats.

9.4 Green Computing and Sustainability:

With the growing energy consumption of data centres, research efforts should aim to improve the energy efficiency and sustainability of cloud computing infrastructures. This includes developing energy-aware resource management techniques, optimizing data centre operations, exploring renewable energy sources for powering data centres, and designing eco-friendly hardware and cooling solutions.

9.5 Serverless Computing and Function as a Service (FaaS):

Serverless computing is gaining popularity as it allows running applications without managing servers or infrastructure. Future research should focus on optimizing serverless architectures, improving resource allocation, and enhancing the scalability and performance of Function as a Service (FaaS) platforms.

9.6 Internet of Things (IoT) and Cloud Integration:

The proliferation of IoT devices generates massive amounts of data that can be processed and analyzed in the cloud. Future research should explore efficient ways to integrate IoT devices with cloud platforms, develop IoT-specific cloud services, and address data storage, security, and real-time analytics challenges.

9.7 Hybrid Cloud Orchestration and Management:

As organizations adopt hybrid cloud environments, research efforts should focus on developing effective orchestration and management frameworks. This includes seamless integration between private and public clouds, workload migration strategies, and unified management interfaces for hybrid cloud deployments.

9.8 Blockchain and Distributed Ledger Technologies in Cloud Computing:

Blockchain technology can enhance cloud computing's trust, transparency, and security. Future research should investigate blockchain integration with cloud services, addressing challenges such as scalability, privacy, and consensus algorithms to enable secure and decentralized cloud deployments.

9.9 Edge Intelligence and Fog Computing:

Edge intelligence leverages the power of edge devices to perform data processing and analysis closer to the data source, reducing latency and bandwidth usage. Future research should focus on developing intelligent edge computing frameworks, optimizing resource management in fog environments, and enabling real-time decision-making at the network edge.

9.10 Data Governance and Compliance in Cloud Environments:

As data regulations become more stringent, future research should explore effective data governance and compliance frameworks for cloud computing. This includes data classification, access control mechanisms, auditing, and accountability in multi-tenant cloud environments to ensure compliance with data protection and privacy regulations.

9.11 Intrusion Detection and Threat Intelligence in Cloud Computing:

With the increasing complexity and sophistication of cyber threats, research efforts should focus on developing advanced intrusion detection and threat

intelligence mechanisms tailored explicitly for cloud computing environments. In addition, the development of intelligent algorithms and machine learning models to detect and mitigate intrusion attempts, as well as integrating threat intelligence feeds and real-time monitoring systems to enhance the security posture of cloud deployments. Furthermore, research should explore using anomaly detection techniques and behavioural analysis to identify and respond to emerging and zero-day threats in cloud environments. By enhancing the capabilities of intrusion detection and threat intelligence in cloud computing, organizations can strengthen their security defences and protect their data and applications from evolving cyber threats. By exploring these future trends and research directions, the cloud computing community can continue to innovate and shape the future of cloud-based technologies, addressing emerging challenges and unlocking new opportunities for organizations across various industries.

10. CONCLUSION:

In conclusion, this paper comprehensively studied cloud computing deployment and service models. It explored the different deployment models, including public, private, hybrid, and community clouds, highlighting their benefits and considerations. The paper also discussed the service models of cloud computing, namely infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), emphasizing their features and advantages.

Through a thorough literature review, the paper presented the background and discussed the latest research and developments in cloud computing. It highlighted the challenges and considerations in cloud deployment, including security, compliance, and data portability.

Furthermore, the paper examined the future trends and emerging technologies in cloud computing, such as edge computing, serverless computing, and multi-cloud strategies. It outlined potential areas of research and development, including AI and ML integration, quantum computing, security enhancements, and green computing.

Overall, this study provides valuable insights into the deployment models, service models, challenges, and future trends in cloud computing. It serves as a foundation for organizations and researchers to understand and explore the potential of cloud computing, enabling them to make informed decisions and contribute to advancing this rapidly evolving field.

REFERENCES:

[1] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... & Zaharia, M. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50-58.

[2] Mell, P., & Grance, T. (2011). The NIST definition of cloud computing. National Institute of Standards and Technology, 53(6), 50.

[3] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 25(6), 599-616.

[4] Vaquero, L. M., Rodero-Merino, L., Caceres, J., & Lindner, M. (2008). A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1), 50-55.

[5] Dillon, T., Wu, C., & Chang, E. (2010). Cloud computing: issues and challenges. In 2010 24th IEEE international conference on advanced information networking and Applications (pp. 27-33). IEEE.

[6] Rittinghouse, J. W., & Ransome, J. F. (2016). Cloud computing: implementation, management, and security. CRC Press.

[7] Subashini, S., & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. Journal of Network and Computer Applications, 34(1), 1-11.

[8] Wang, L., von Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J., ... & Fu, C. (2018). Cloud computing: a perspective study. New Generation Computing, 36(4), 313-345.

[9] Hamdaqa, M., Sahandi, R., & Asim, M. (2018). A survey of cloud service models. Future Generation Computer Systems, 78, 535-550.

[10] Rimal, B. P., Jukan, A., & Katsaros, D. (2018). An overview of service models in cloud computing.

Journal of Network and Computer Applications, 67, 106-127.

[11] Ristenpart, T., Tromer, E., Shacham, H., & Savage, S. (2009). Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In Proceedings of the 16th ACM conference on Computer and communications security (pp. 199-212).

[12] Liang, X., Lu, R., & Yang, L. T. (2016). A comprehensive study on security of cloud computing. In IEEE transactions on parallel and distributed systems, 27(2), 478-490.

[13] Botta, A., et al. (2016). Integration of Cloud computing and Internet of Things: A survey. Future Generation Computer Systems, 56, 684-700. https://doi.org/10.1016/j.future.2015.09.021

[14] Almorsy, M., Grundy, J., & Müller, I. (2016). An analysis of the cloud computing security problem. arXiv preprint arXiv:1609.01107.

[15] Rimal, B. P., Choi, E., & Lumb, I. (2018). A taxonomy and survey on autonomic management of applications in cloud computing. IEEE Communications Surveys & Tutorials, 20(1), 674-711.

[16] Hassan, M. M., Zhang, H., Nasser, Y., & Al-Salman, A. (2018). A hybrid cloud architecture for big data analytics. IEEE Access, 6, 24857-24867.

[17] Liu, J., Liu, C., Chen, S., Chen, C., & Ning, H. (2020). A cooperative game theory based resource allocation in community cloud computing. Future Generation Computer Systems, 102, 287-297.

[18] Zhang, Q., Zhang, Z., & Zhang, Q. (2019). An intrusion detection system for cloud computing based on hierarchical deep belief network. Future Generation Computer Systems, 92, 214-224.

[19] Rauti, S., Zavarsky, P., Stavrou, A., & Nucci, A. (2020). Cyber threat intelligence for cloud computing security: Review, potential, and challenges. Journal of Network and Computer Applications, 170, 102798. doi:10.1016/j.jnca.2020.102798

[20] Zheng, R., Li, Z., Zhou, Q., & Zhou, W. (2021). Security challenges and mitigation strategies in cloud computing: A comprehensive survey. Future Generation Computer Systems, 118, 627-647.

[21] Teng, F., Yu, S., & Li, H. (2020). A comprehensive survey on cloud deployment models. Future Generation Computer Systems, 108, 347-359.

[22] Almorsy, M., Grundy, J., & Ibrahim, A. (2021). A comprehensive review of service models in cloud computing. Journal of Systems and Software, 179, 110911.

Journal of Intelligent System and Applied Data Science (JISADS)

Journal homepage : https://www.jisads.com

ISSN (2974-9840) Online

Machine Learning-Based Mobile Payment System for Empowering Low-Income Earners in India

Cleodine W Chiome[1] *, Abdul Basit Darem*[2]*

Basit.darem@yahoo.com

[1]PG Dept Of CS, St. Philomena's College, Mysore, India

[2]Northern Border University, KSA.

Abstract

In the contemporary era, mobile phones have become indispensable, serving as communication devices and offering myriad applications. Among these applications, mobile payment systems hold great potential in driving the transition to a cashless economy. However, India's current mobile payment landscape is limited by its dependency on bank accounts. This poses a significant challenge for low-income earners who lack access to banking services or perceive minimal benefits in owning an account due to their limited financial resources. As a result, they are unable to leverage existing mobile payment systems. Despite bank branches in approximately 40,000 out of 600,000 Indian villages, a staggering 80% of households possess mobile phones. The heavy reliance on cash-based transactions leaves India's economy susceptible to disruptions. To address this, the proposed research project aims to develop a user-friendly and secure mobile payment system that operates independently of bank accounts, ensuring inclusivity for all individuals, irrespective of their banking status.

Moreover, the system will be compatible with any mobile phone and will not require an internet connection. By implementing such an innovative system, India can make significant strides towards achieving a truly cashless economy. Additionally, integrating machine learning techniques in the system can enhance security, fraud detection, and user personalization, further optimizing the user experience and driving the adoption of mobile payments.

Keywords: communication, payment system, machine learning, user experience of mobile payments

1. INTRODUCTION

Mobile communications technology has quickly become the world's most common way of transmitting voice, data, and services in the developing world. They carry the potential to be the best media for the dissemination of information because mobile services are widely available and inexpensive [1]. Mobile phones have been proven to provide reliable access to information for people in low- and mid-income countries, where other forms of communication perform poorly. As a result of the widespread adoption of mobile phones, there has been an increase in the number of Mobile Applications (M-Services) used as a tool for disseminating different types of information to people [2].

Mobile payment is using mobile devices, such as mobile phones, to facilitate payment transactions. Mobile devices can be used for both proximity and remote payments. Mobile payment systems have

increased significantly, to the point where a cashless world is possible, including in India. According to a source, worldwide mobile commerce revenues amounted to 96.34 billion U.S. dollars in 2015 and are set to surpass 693 billion U.S. dollars in 2019 [3][4]. This vast increase in mobile commerce revenues demonstrates the global adaptation to mobile-related services.

Mobile payment systems incorporate a combination of technological innovations developed throughout mobile evolution. These innovations include messaging-based payment services, such as short message (SMS) and multimedia message (MMS)-initiated payments, stored value-based payment services like mobile wallets and accounts, and mobile identification and authorization-based payment services that utilize secure wireless identification modules (SWIM/WIM) along with wireless public key infrastructure (WPKI/PKI) or other identification and authorization schemes for digital signatures and certificates in high-value payment transactions [5].

Looking at a worldwide perspective, mobile applications and services have increased, providing payment solutions for real-time payments, such as in restaurants, shops, vending machines, ticketing, the purchase of mobile services, mobile commerce (applications, software, mobile games), electronic banking, online banking, and peer-to-peer transfers [6].

Several mobile payment systems currently exist, utilizing different financial payment methods, including cryptocurrencies like Bitcoin, direct debit, credit cards, and payment against service bills [7].

The cash crisis in India has been a significant issue since the demonetization of old currency notes, leading to a cash shortage and impacting the economy and daily transactions. This problem particularly affects low-income earners relying heavily on cash-based transactions for their livelihoods, including informal businesses like vendors, rickshaw owners, and farmers. Many individuals lack access to bank accounts, especially in rural areas where banks and ATMs are scarce. The limited penetration of banking services in villages exacerbates the cash crisis, with a shortage of ATMs and entire villages lacking banking facilities [8].

To address these challenges, integrating machine learning (ML) technologies can offer potential solutions. ML algorithms can help analyse transaction patterns and user behaviour to develop models that optimize cash flow, predict demand, and manage supply. Such models can aid in ensuring the availability of cash at ATMs, reducing instances of dry ATMs and providing greater access to cash for low-income earners. Additionally, ML can identify areas with higher cash demands and optimize the deployment of ATMs or other cash dispensing services to meet the population's needs.

Furthermore, ML algorithms can contribute to developing mobile payment systems that cater to low-income earners without bank accounts. These systems can leverage alternative authentication methods, such as biometrics or unique identification numbers, to enable secure and inclusive mobile payment transactions. ML-based fraud detection techniques can also enhance the security of these systems, mitigating risks associated with digital transactions [9].

By integrating ML technologies into cash management and mobile payment systems, the challenges faced by low-income earners in accessing and utilizing cash can be addressed more effectively. These advancements can potentially contribute to India's more inclusive and resilient financial ecosystem.

2. LITERATURE REVIEW

Mobile payment systems have gained significant attention in recent years, revolutionizing how people transact. With the proliferation of smartphones and technological advancements, mobile payment systems have become an integral part of our daily lives. This part aims to provide a comprehensive overview of the existing research and developments in mobile payment systems. It focuses on their impact on low-income earners in India and the integration of machine learning (ML) techniques.

Mobile Payment Systems and Financial Inclusion:

Mobile payment systems can potentially address financial inclusion challenges faced by low-income earners in India. Research by [10] highlights the significance of mobile phones in providing reliable access to financial services in low and middle-income countries. By leveraging mobile payment systems,

individuals without bank accounts can still participate in the digital economy, making transactions and accessing financial services more easily.

Challenges Faced by Low-Income Earners:

Low-income earners in India often encounter difficulties accessing traditional banking services due to limited infrastructure and low bank penetration in rural areas. According to the World Bank, access to banking services remains challenging for millions in India. These challenges hinder financial inclusion and limit the ability of low-income earners to engage in digital payment systems.

Machine Learning in Mobile Payment Systems:

Integrating machine learning techniques in mobile payment systems offers numerous opportunities for improving security, personalization, and fraud detection. ML algorithms can analyze transaction patterns, detect anomalies, and accurately identify fraudulent activities. For example, [11] explore using ML and user behaviour analysis to enhance mobile payment security and detect suspicious activities in real-time.

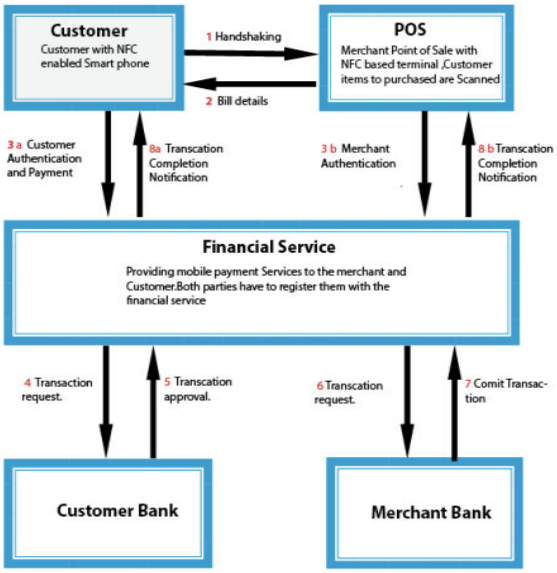

Figure 2.1 NFC Mobile Payment System

Adoption and Acceptance of Mobile Payment Systems:

The widespread adoption and acceptance of mobile payment systems are essential for their success. Several factors influence user adoption, including trust, convenience, security, and perceived usefulness. A study by [12] highlights the importance of user trust and security concerns in influencing adoption behaviour. Ensuring the security and privacy of user data is crucial in building trust among low-income earners and encouraging their adoption of mobile payment systems.

User Experience and Design Considerations:

The user experience plays a vital role in the success of mobile payment systems. A user-friendly interface, simple design, and ease of use are critical factors for encouraging adoption among low-income earners. Studies by [14] emphasize the need for intuitive interfaces and seamless user experiences to enhance the acceptance and usability of mobile payment systems.

The literature reviewed demonstrates the significant potential of mobile payment systems in promoting financial inclusion among low-income earners in India. Integrating machine learning techniques in mobile payment systems can enhance security, detect fraud, and improve user experience. However, challenges such as limited infrastructure, trust, and security concerns must be addressed to ensure low-income earners' successful adoption and usage of mobile payment systems. Future research should focus on developing innovative solutions that address these challenges and promote the widespread adoption of mobile payment systems among underserved populations.

3. METHODOLOGY AND SYSTEM STRUCTURE

The proposed mobile payment system aims to leverage machine learning (ML) techniques to create a simple and secure solution that can be used with any type of mobile phone without requiring an internet connection. The system will utilize USSD technology for communication. It will not be directly connected to the customer's bank account, making it accessible to holders and non-account holders.

One key aspect of the proposed system is integrating digital services with payment-on-site terminals at grocery shops and fuel stations. This feature eliminates the need for physical wallets and loose currency, providing convenience to users.

Additionally, the system will enable person-to-person money transfers. The primary focus is to assist low-income earners in their daily transactions, but the system can be used by anyone, regardless of income level, location, or mobile device type.

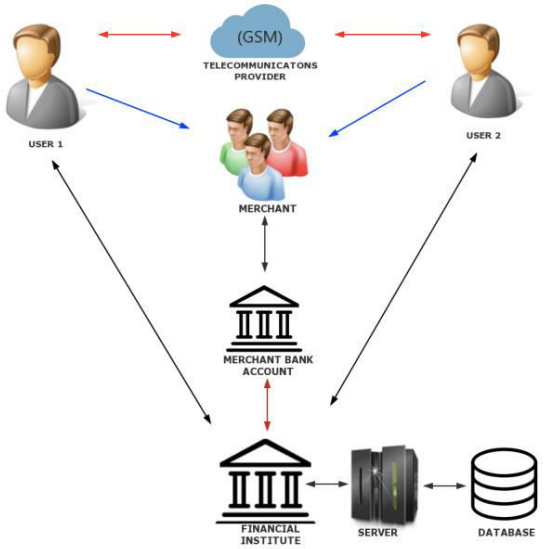

Figure 2. System Architecture

The project's objectives include designing a user-friendly mobile payment system that requires no prior training and is compatible with any mobile phone or tablet. The system should prioritize speed and efficiency to eliminate queues and delays during transactions, ensuring user satisfaction. Security is paramount, with robust measures implemented to protect user information and prevent unauthorized access.

Moreover, the system aims to be highly accessible, allowing users to make payments anytime and anywhere. Unlike traditional banks with limited operating hours, the mobile payment system will be available 24/7. It should also seamlessly integrate with various payment terminals, enabling transactions in diverse settings such as shops, malls, supermarkets, hotels, restaurants, and public transportation.

Lastly, the system emphasizes safety, reliability, and error prevention. The ML techniques incorporated will enhance the system's capabilities in detecting and preventing fraudulent activities, ensuring user trust and confidence.

GSM (Global System for Mobile Communications, originally Groupe Spécial Mobile) is a standard developed by the European Telecommunications Standards Institute (ETSI) to describe the protocols for second-generation (2G) digital cellular networks used by mobile phones, first deployed in Finland in July 1991. GSM is a circuit-switched system that divides each 200 kHz channel into eight 25 kHz timeslots. GSM operates on the mobile communication bands 900 MHz and 1800 MHz in most parts of the world. In the US, GSM operates in the bands 850 MHz and 1900 MHz.

GSM makes use of narrowband Time Division Multiple Access (TDMA) technique for transmitting signals. GSM was developed using digital technology. It has the ability to carry 64 kbps to 120 Mbps of data rates. Despite the ongoing development of 5G, the already existing third generation (3G) UMTS standards developed by the 3GPP and fourth-generation (4G) LTE advanced standards; GSM technology is still the backbone of mobile communications. Since 2014 it has over 90% of its market share, operating in over 219 countries and territories with more than one billion users.

The proposed mobile payment system architecture described above comprises four modules: cash in, cash out, person-to-person, and customer-to-merchant.

The architecture includes the following components:

Users/Customers: Users must register with their mobile numbers to access the mobile wallet. They can transfer funds between users via USSD codes over the GSM network. Users can also make payments at merchants' shops using their mobile phones.

GSM (Global System for Mobile Communications): This project works with GSM-supported mobiles and integrates with all mobile types through USSD (Unstructured Supplementary Service Data), a communication protocol used for sending text messages between a mobile phone and an application server.

Merchant: Merchants, including shop owners and business owners, can receive user payments through the mobile payment system. The funds received by merchants are automatically deposited into their bank accounts.

Merchant Bank Account: The merchant bank account holds the funds received from users when they

purchase goods and services from merchants through the mobile payment system.

Financial Institution: A financial institution is responsible for storing and managing users' funds, eliminating the need for users to have a traditional bank account. The financial institution securely holds the users' money.

Web Server: The web server hosts the system's databases, which store and manage users' account information. The financial institution uses the webserver to handle user accounts and transactions.

Database: The database is an organized collection of data that supports storing and retrieving information. In the mobile payment system, it stores schemas, tables, queries, reports, and other objects related to user accounts and transaction data.

The system enables users to utilize the mobile payment system using GSM-supported mobiles associated with mobile numbers linked to SIM cards. Users can transfer funds using USSD, compatible with any mobile phone. User accounts are password protected. Merchants receive payments through the mobile payment system, and the funds are directly deposited into their bank accounts.

4. DISCUSSION

Mobile payment systems have been in existence in India for some time now, with numerous systems currently being utilized. These systems represent a significant step towards achieving a cashless economy in India. However, the existing mobile payment systems are all linked to bank accounts, making them inaccessible to individuals who do not possess a bank account. In India, there is a substantial population residing in rural areas where banking facilities are scarce. Many villages lack banks altogether, while others have only a single bank branch and a shared ATM serving a population of over 100,000 people [15]. As a result, a significant portion of the rural population does not utilize formal banking services.

According to the Global Findex survey, 43% of Indian bank account holders have inactive accounts, while others maintain zero balances. These statistics hinder the growth of mobile payment systems in India, as they demonstrate the limited adoption and utilization of banking services among the population.

To address these challenges and promote the growth of mobile payments, this project aims to develop a mobile payment system that accommodates low-income earners without access to traditional banking services. The system is designed to be compatible with any type of mobile device, whether it is a smartphone or a basic feature phone, using USSD (Unstructured Supplementary Service Data) technology. It enables fund transfers between individuals and facilitates payments at various establishments such as shops, supermarkets, malls, and cinemas. Moreover, the system has the capability to integrate with existing registered systems, ensuring widespread acceptance and interoperability. By eliminating the need for physical cash transactions, the system helps alleviate the impact of cash shortages on the economy.

5. CONCLUSION

The proposed mobile payment system for low-income earners in India combines mobile technology with machine learning techniques to create a simple and secure solution for everyday transactions. By leveraging machine learning algorithms, the system enhances security, detects fraudulent activities, and personalizes user experiences. With compatibility across any mobile device and USSD technology, the system ensures accessibility without requiring an internet connection.

This innovative approach addresses the challenges individuals face without access to traditional banking services, promoting financial inclusion and contributing to the advancement of a cashless economy. Future enhancements can extend the system's capabilities, such as enabling online shopping, facilitating international fund transfers, and providing personalized user recommendations.

Successful implementation of the mobile payment system relies on collaboration among businesses, merchants, and consumers, along with establishing a robust regulatory framework and widely accepted standards. By leveraging the transformative potential of machine learning, this system has the capacity to revolutionize financial practices and drive economic development in India.

REFERENCES

[1] J. Muthee and N. Mhando, "African Media Development Initiative Tanzania," 2006

[2] Wikipedia (2008) Short Message Service. Available: http://en.wikipedia.org/wiki/SMS#cite_note-1

[3] https://www.statista.com/statistics/557951/mobile-commerce-transaction-value worldwide/

[4] GSMA. (2019). State of the industry report on mobile money. Retrieved from https://www.gsma.com/mobilefordevelopment/wp-content/uploads/2019/04/State-of-the-Industry-Report-on-Mobile-Money_2019.pdf

[5] Ling, R., & Donner, J. (2009). Mobile communication. Polity Press.

[6] Arroyo, E., & Mendoza, L. (2014). Mobile money: Overview and analysis of technology adoption and usage patterns. The Journal of Technology Studies, 40(1), 35-46.

[7] Singh, R., & Sachdeva, P. (2019). Mobile payment systems: A review of technologies and business models. International Journal of Advanced Research in Computer Science and Software Engineering, 9(7), 166-174.

[8] The News Minute. (n.d.). No ATM for 25km: Villages in Madhya Pradesh struggle with demonetisation. Retrieved from https://www.thenewsminute.com/article/no-atm-25km-villages-madhya-pradesh-struggle-demonetisation-52903

[9] Choudhury, D., & Barua, S. (2020). Design and Implementation of Mobile Payment System Using Machine Learning. International Journal of Engineering and Advanced Technology (IJEAT), 9(5), 2799-2804.

[10] Donner, J. (2008). Research approaches to mobile use in the developing world: A review of the literature. The Information Society, 24(3), 140-159.

[12] Li, L., & Yu, Z. (2018). Mobile payment authentication based on machine learning and user behavior analysis. IEEE Access, 6, 47379-47386.

[13] Gupta, A., & Dalal, U. (2020). A study on factors influencing mobile payment adoption and user behavior. International Journal of Business Innovation and Research, 21(2), 255-273.

[14] Choudhury, D., & Barua, S. (2020). Design and implementation of mobile payment system using machine learning techniques. International Journal of Innovative Technology and Exploring Engineering, 9(5), 2799-2804.

[15] http://www.cab.org.in/FILCPortal/Lists/Implementations/Attachments/10/operational_manual_financial.pdf

Journal of Intelligent System and Applied Data Science (JISADS)

Journal homepage : https://www.jisads.com

ISSN (2974-9840) Online

API Malware Analysis: Exploring Detection and Forensics Strategies For Secure Software Development

Husam Alalloush[1], Wasim A. Ali[1]*

husamalalloush@gmail.com, wasim.ali@poliba.it

[1]Chaitanya University, India, [2]Politecnico di Bari, Italy

Abstract

API Malware Analysis and Forensics is a key field of research in cybersecurity. It is critical to have strong defenses in place to detect and prevent malware attacks. APIs, since they can have disastrous consequences. The article aims to provide a thorough overview of the current state of the art in API malware analysis and forensics, as well as the methods and equipment used to discover, analyses, and combat API-based malware assaults. Also covered will be an overview of the various approaches for identifying malware in APIs, such as static and dynamic analysis. The primary purpose of this work is to offer a comprehensive overview of API malware analysis and investigation, spanning numerous approaches and instruments used to detect and investigate API malware. This study also emphasizes the importance of taking proactive steps to prevent API-based malware attacks, such as testing APIs for vulnerabilities regularly, implementing security protocols, and deploying cutting-edge security technologies to detect and mitigate API-based malware attacks.

1. INTRODUCTION

An application programming interface (API) is a collection of protocols, procedures, and tools that allows software developers to communicate and collaborate. APIs provide a standard method for exchanging data and services across multiple software components, irrespective of the underlying hardware and operating systems [1].

APIs play a crucial role in creating modular and scalable applications in software development. Using APIs, developers can deconstruct complex systems into smaller, independent components that can be independently developed, evaluated, and deployed. APIs also enable developers to utilise existing code and services, which saves time and reduces development costs [2]. APIs can be utilised in numerous software applications, including web applications, mobile apps, and desktop software. Typically, they connect disparate software systems, such as a front-end web application, to a back-end database [3].

However, APIs can also represent a potential security risk. Malicious actors can exploit API design and implementation vulnerabilities to initiate API malware

attacks. Therefore, developers and security professionals must implement appropriate security measures and undertake regular API security testing to ensure the integrity of their systems [4].

2. API MALWARE ATTACKS: A REAL DANGER

API security has distinct characteristics that differentiate it from traditional security. Firstly, APIs introduce a challenge because they employ various protocols and multiple endpoints, unlike traditional networks that mainly focus on protecting specific ports like HTTP (port 80) and HTTPS (port 443). As APIs evolve, even a single API can become a complex security task [3].

Secondly, APIs in a DevOps context often undergo frequent changes, making it difficult for traditional security tools, such as Web Application Firewalls (WAFs), to handle their elasticity. These tools require manual tuning and reconfiguration whenever an API changes, which is prone to errors and consumes valuable resources and time [3].

Thirdly, clients accessing APIs are not limited to web browsers. Native and mobile applications and other services and software components often interact with service or microservice APIs. Traditional web security technologies relying on browser verification cannot effectively identify harmful bots in automated traffic originating from API endpoints, as these clients do not utilize browsers [3]. It's important to note that examining incoming requests alone does not guarantee the detection of attacks since many API abuse attacks can mimic legitimate requests.

3. THREAT OF API MALWARE ATTACKS API

API malware attacks pose a significant threat in the realm of cybersecurity. These attacks utilize APIs to inject and execute malicious code on a targeted system. The malware is often concealed within API calls, exploiting vulnerabilities to gain unauthorized access or control over the system. API malware attacks can manifest in various ways, including remote code execution, credential theft, data exfiltration, and DDoS attacks [22][10][12].

Remote code execution involves injecting malware through an API call, enabling attackers to execute code on the targeted system remotely. Credential theft occurs when malware is employed to pilfer user credentials through API calls, such as usernames and passwords. Data exfiltration involves extracting sensitive data from the targeted system using API calls. Additionally, through APIs, malware can initiate Distributed Denial of Service (DDoS) attacks, inundating the targeted system with excessive traffic and disrupting normal operations.

To mitigate the risk of API malware attacks, developers and security professionals must implement robust security measures and regularly conduct API security testing.

4. API MALWARE ANALYSIS AND FORENSICS: A CRUCIAL FIELD OF STUDY

API malware analysis and forensics play a critical role in detecting, analyzing, and mitigating the impact of API malware attacks. These attacks can lead to severe consequences for organizations, including data breaches, system downtime, financial losses, and damage to their reputation [10][22]. Conducting effective API malware analysis and forensics is crucial in identifying the source and extent of the attack, recovering lost or stolen data, and implementing measures to prevent future attacks.

API malware analysis involves examining APIs and their associated code to identify indicators of malware, such as unusual API calls or unexpected system behavior. Detecting malware can be challenging since it may be disguised or obfuscated to evade detection. On the other hand, API malware forensics involves conducting a thorough investigation of the attack to determine its root cause and develop preventive measures against similar attacks in the future. This process may include analyzing system logs, studying network traffic, and examining other digital evidence to reconstruct the attack and assess the extent of the damage [22].

The significance of API malware analysis and forensics has grown in parallel with the increasing use of APIs in software development. As more organizations rely on APIs to connect their systems and services, the potential attack surface for API malware attacks has also expanded [22][10]. In conclusion, organizations must prioritize API malware analysis and forensics to safeguard against the detrimental effects of API malware attacks. By investing in these practices, organizations can uphold the integrity and security of their APIs and proactively prevent future attacks [22].

5. TYPES OF API MALWARE ATTACKS

Organizations should be aware of various common types of API malware attacks that pose a risk to their systems [8][9][12]. These attacks include:

API Spoofing: Attackers create fake APIs that imitate legitimate ones. When users connect to these fake APIs, attackers can steal user credentials or inject malware into the user's system.

API Injection: Malicious code is inserted into valid API calls to execute it on the targeted system. This can be achieved by exploiting API input flaws or intercepting and modifying API calls using man-in-the-middle attacks.

API Parameter Tampering: Attackers modify parameters in API calls to gain unauthorized access or manipulate data. This can be done by intercepting and modifying API calls or using automated tools to manipulate API inputs.

API Denial-of-Service (DoS) Attacks: APIs are overwhelmed with excessive requests, causing them to crash or become unresponsive. This can be achieved by flooding the API with requests using automated tools or exploiting vulnerabilities in the API's design or implementation.

API Phishing: Users are deceived into connecting to fake APIs that appear legitimate. When users enter their credentials into these fake APIs, attackers steal them for future use.

API Remote Code Execution (RCE): API RCE attacks leverage weaknesses in APIs to execute arbitrary code on the targeted machine. This can be accomplished by using a malicious payload in an API call or exploiting vulnerabilities in the API's input validation or authentication mechanisms.

The number of APIs deployed within organizations is rapidly increasing, with a survey showing that 26% of businesses now use at least twice as many APIs compared to the previous year. This surge in API usage has made APIs a prime target for attacks [9]. It is crucial for organizations to be aware of these types of API malware attacks and implement appropriate security measures to protect their systems and data from potential vulnerabilities and unauthorized access.

6. EXPLANATION OF HOW API MALWARE CAN BE USED TO EXECUTE MALICIOUS CODE

API malware can exploit vulnerabilities in software components that utilize APIs to execute malicious code [13][14]. Attackers hide malware within API calls, enabling them to inject and execute malicious code on a targeted system. One common method is remote code execution (RCE), where attackers send a payload containing malicious code through an API. This payload is executed on the system, granting the attacker remote access and control [13][14]. Another technique is API injection, where attackers inject malicious code into legitimate API calls, taking advantage of API input flaws or intercepting and modifying API calls through man-in-the-middle attacks [22]. API malware can also execute malicious code through credential theft, data exfiltration, and DDoS attacks. For instance, API malware can steal user credentials through API calls and subsequently utilize those credentials to execute malicious code on the targeted system [15][10].

To protect against API malware attacks that execute malicious code, organizations should implement secure API design, authentication and authorization mechanisms and monitor API activity for suspicious behaviour. Regular API security testing and analysis can also help detect and prevent API malware attacks [10][22].

Examples of real-world API malware attacks:

Facebook API Malware Attack: In 2018, attackers exploited an API vulnerability on Facebook to steal access tokens and compromise more than 30 million user accounts. The attack leveraged the "View As" feature to access and control user accounts [4].

Twitter API Malware Attack: In 2013, a malware attack on Twitter's APIs resulted in the theft of user data, including passwords and email addresses. Attackers exploited a cross-site scripting (XSS) flaw in Twitter's mobile app [5].

Uber API Malware Attack: In 2016, attackers targeted Uber's APIs, compromising the personal data of over 57 million users and drivers. The attack exploited an API vulnerability to gain unauthorized access to a database, which was then downloaded and encrypted [6].

Salesforce API Malware Attack: In 2018, a malware attack on Salesforce's APIs led to the theft of customer data from multiple Salesforce customers. Attackers exploited an API vulnerability to access customer data, using it for phishing attacks and other fraudulent activities.

Equifax API Malware Attack: In 2017, a malware attack on Equifax's APIs exposed personal data belonging to over 143 million customers. Attackers exploited an API vulnerability to access customer data, which was downloaded and exfiltrated [7].

These real-world examples highlight the damaging consequences of API malware attacks and emphasize the importance of robust API security measures to safeguard sensitive data and prevent unauthorized access.

7. TECHNIQUES FOR API MALWARE DETECTION

API malware can be detected using various methods, including signature-based detection, behaviour-based detection, and machine learning-based detection [2][1].

Signature-based detection: This method involves searching for patterns or signatures of known malware within API requests. Signatures, which are derived from well-known malware, are used to identify related malware in API calls. While effective against known malware, signature-based detection may fail to detect new or undiscovered threats.

Behaviour-based detection: This approach focuses on analyzing the behaviour of API calls to detect potential malware. Behaviour-based detection involves creating a baseline by profiling normal API call behaviour and then identifying any deviations from the baseline. This method can detect new and unknown malware, but it may also produce false positives.

Machine learning-based detection: In this method, machine learning techniques identify abnormal patterns in API calls. Machine learning algorithms are trained on typical API call behaviour to detect deviations from the norm. This approach can detect brand-new and unidentified malware but may also result in false positives and negatives.

The advantages and disadvantages of signature-based and machine learning-based, techniques are summarized in Table 1.

Table 1. signature-based and machine learning-based advantages and disadvantages.

Feature	Signature-based	Machine Learning-based
Advantage	Reduced runtime, Easy to implement	More effective in finding polymorphic malware, Can detect unknown malware
Disadvantage	Unknown malware cannot be detected, Requires regular updates	Requires a significant amount of labeled training data, Can be computationally expensive
Accuracy	High, Low	High, Can be high or low depending on the model
False positives	Low, High	Low, Can be high depending on the model
False negatives	High, Low	High, Can be low depending on the model

8. TECHNIQUES OF API MALWARE ANALYSIS.

1- Static Analysis: Static analysis involves examining the code within API calls without executing it. This technique typically relies on automated tools to scan the code for known malicious patterns, vulnerabilities, or code obfuscation techniques. It analyzes the code's structure, syntax, and content to identify potential security issues. Static analysis tools may use pattern matching, rule-based analysis, or abstract interpretation to detect known malware signatures or suspicious code constructs. However, static analysis may struggle with detecting sophisticated or previously unseen malware as it relies on pre-existing knowledge of known patterns.

2- Dynamic Analysis: Dynamic analysis involves executing API calls in a controlled environment to observe their behavior and interactions. It captures runtime information and monitors network traffic, system calls, memory usage, and other runtime characteristics. By analyzing the behavior of API calls during execution, dynamic analysis can identify abnormal or malicious activities, such as unauthorized

data access, privilege escalation, or suspicious network communications. Dynamic analysis can provide insights into runtime code execution, data flow, and interactions with the underlying system. It effectively detects behavior-based attacks and identifying unknown or zero-day threats that may evade static analysis. However, dynamic analysis can be resource-intensive and time-consuming, especially when dealing with large-scale or complex systems.

3- Sandboxing: Sandboxing involves running API calls in an isolated and controlled environment known as a sandbox. The sandbox provides a virtualized or containerized environment that emulates the necessary system resources and dependencies to execute API calls safely. By isolating the execution of API calls, sandboxes prevent potential damage to the underlying system. Sandboxing allows analysts to observe the behavior of API calls in a controlled environment, monitoring system interactions, file system modifications, network communications, and other runtime activities. It helps identify potentially malicious behaviors or activities that might harm the host system. However, advanced malware may be designed to evade sandbox detection by detecting the presence of a sandbox environment or by employing techniques to delay malicious activities.

4- Memory Forensics: Memory forensics involves analyzing a system's volatile memory (RAM) to gather evidence and extract information related to security incidents or malicious activities. In analyzing API calls, memory forensics can provide valuable insights into runtime behavior, data structures, and potential code injections or modifications performed by malware. By examining the memory space used by an application or API, analysts can uncover artefacts, such as injected code, hooks, or altered data, that may indicate the presence of malicious code. Memory forensics can also help identify malware persistence mechanisms or uncover encryption keys and passwords used by the malicious code. Including memory forensics in API call analysis can enhance the depth of investigation and aid in detecting advanced or memory-based attacks.

5- API Fuzzing: API fuzzing is a technique used to test the robustness and security of APIs by sending a large volume of malformed or unexpected inputs to an API and monitoring its response. The goal is to identify vulnerabilities or weaknesses in the API implementation that attackers could exploit. By fuzzing API inputs, analysts can uncover security flaws, such as buffer overflows, injection vulnerabilities, or error-handling issues that might lead to unauthorized code execution or other forms of API abuse. While API fuzzing is primarily used for testing and security assessment, it can indirectly aid in identifying potential malicious code injection points or vulnerabilities within API calls. Incorporating API fuzzing as part of the analysis can help identify weaknesses and harden the security of APIs.

Combining these techniques is often employed for comprehensive API call analysis and identifying malicious code. Static analysis is useful for quickly identifying known patterns and vulnerabilities, while dynamic analysis provides a deeper understanding of runtime behavior. Sandboxing offers a controlled environment for executing and observing API calls. These techniques are often complemented with other security measures, such as threat intelligence, anomaly detection, and continuous monitoring, to enhance the overall effectiveness of API call analysis and mitigate the risk of malicious code execution.

9. 4.API FORENSICS

API forensics is the process of investigating and examining APIs to determine if they have been exploited, misused, or compromised. It involves applying forensic techniques and technologies to uncover security flaws, gather evidence for legal purposes, and collect relevant data from both the APIs themselves and the systems connected to them. API forensics plays a crucial role in today's interconnected world, where systems and platforms heavily rely on APIs for seamless integration [16][20].

The significance of API forensics stems from the increasing reliance on APIs to enable communication and data exchange between various systems, services, and applications. APIs serve as the interface for these interactions, making them an attractive target for attackers seeking to exploit vulnerabilities or gain unauthorized access. By conducting API forensics, investigators can thoroughly analyze the APIs and associated systems to identify any signs of compromise, abuse, or security breaches.

API forensics involves several key activities and techniques. These may include:

Data Collection: Gathering relevant information and data from the APIs and the systems they connect to. This includes obtaining API logs, network traffic data, server logs, and any other available artifacts that may hold evidence of malicious activity or security incidents.

Traffic Analysis: Analyzing the network traffic generated by the API calls, including request and response data. This analysis can help identify anomalous patterns, suspicious activities, or unauthorized access attempts.

Code Review: Reviewing the API code, including the endpoints, authentication mechanisms, input validation, and error handling. This examination aims to identify any vulnerabilities, insecure coding practices, or potential attack vectors that could be leveraged by malicious actors.

API Access and Usage Analysis: Examining access controls, authentication mechanisms, and usage patterns of the APIs. This analysis helps identify any unauthorized access, abnormal usage patterns, or misuse of the APIs.

Incident Reconstruction: Reconstructing the sequence of events leading up to a security incident or compromise involving the APIs. This involves analyzing various artifacts, such as logs, timestamps, and system states, to understand the timeline and the methods used by attackers.

Digital Evidence Preservation: Ensuring the proper preservation and integrity of digital evidence collected during the API forensic investigation. This is crucial for maintaining the admissibility and reliability of the evidence in potential legal proceedings.

API forensics is valuable not only for detecting and mitigating security incidents but also for supporting legal actions. The evidence collected during API forensic investigations can be used in court cases or internal disciplinary actions to hold perpetrators accountable, establish liability, or prove compliance violations.

As the reliance on APIs continues to grow, the importance of API forensics becomes even more significant. By applying forensic techniques and leveraging appropriate technologies, organizations can ensure the integrity, security, and trustworthiness

of their API ecosystems and respond effectively to potential security incidents.

10. API FORENSICS PROCESS

The API forensics process consists of several essential steps, each contributing to the comprehensive analysis of APIs and the identification of security breaches and attacks. Here is an expansion of each step:

Identification: The first step involves identifying the APIs utilized within the organization's systems and platforms. This includes understanding the types of APIs being used, their specific functionalities, and the systems they are connected to. It is crucial to have a clear overview of the API landscape to focus the forensic investigation effectively.

Collection: Once the APIs have been identified, the next step is to collect relevant data from these APIs. This includes gathering API logs, network traffic data, server logs, and any other available information that can aid in identifying security breaches and attacks. The collection process should aim to capture a comprehensive dataset that covers the period of interest.

Analysis: The collected data is then subjected to analysis using forensic techniques and specialized tools. This involves examining traffic patterns, analyzing request and response data, identifying anomalies or abnormal behaviors, and tracing potential attack vectors. The analysis helps in understanding the scope and impact of security breaches, as well as determining the root cause of any malicious activities.

Evidence Gathering: In API forensics, the focus is on gathering evidence that can be utilized in legal proceedings or internal disciplinary actions. This step involves preserving the collected data in a secure manner to maintain its integrity and admissibility as evidence. Proper documentation, including timestamps, metadata, and chain of custody, should be established to create an audit trail of the investigation. The findings and conclusions of the API forensics analysis should also be documented as part of the evidence-gathering process.

It's important to note that the API forensics process may vary depending on the specific requirements, available resources, and the nature of the investigation. It may involve additional steps, such as incident reconstruction or collaboration with legal professionals. Furthermore, throughout the process, adherence to best practices and legal requirements, including data protection and privacy regulations, is crucial.

By following a systematic API forensics process, organizations can effectively identify and respond to security breaches, collect valuable evidence for legal purposes, and improve the overall security posture of their API ecosystems.

11. CHALLENGES OF API FORENSICS

API forensics encounters various challenges that impede its effectiveness. One primary obstacle is the complexity inherent in modern API architectures, which often involve multiple levels and components. This complexity poses difficulties in identifying and isolating security flaws and attacks. Furthermore, the forensic investigation process is further complicated by the reliance on third-party services as the foundation for APIs, adding intricacy to the analysis [19][21].

Another challenge is the absence of standardized forensic methods and tools tailored for API forensics. While certain tools are available, they are often

proprietary and not widely adopted. This lack of standardized tools hinders collaboration and data sharing among researchers, making it challenging to collaborate and effectively leverage collective expertise in API forensics [19][21].

In summary, the challenges of API forensics include the complexity of contemporary API architectures with multiple levels and components, as well as the dependence on third-party services. Additionally, the absence of standardized forensic methods and tools further complicates the forensic investigation process, hindering collaboration and data sharing among researchers. These challenges highlight the need for continued research and development to address these complexities and enhance the effectiveness of API forensics.

12. TECHNIQUES FOR API FORENSICS

API forensics encompasses retrieving, preserving, and analysing evidence about attacks targeting APIs. The following techniques are commonly employed in API forensics [16][18][19][20][21]:

Memory Dump Analysis: This technique involves extracting the memory from a machine to identify any malicious activity. Memory dump analysis is useful for locating malware that may not exist in the file system but resides solely in memory. Specialized tools and methods, such as the Volatility Framework, can be utilized to analyze memory dumps effectively.

Log Analysis: Log analysis examines system logs for suspicious activities associated with API-based attacks. System logs can provide crucial details about the behaviour of API calls, including timestamps, source and destination addresses, and the nature of the operations performed. Both automated tools like Log Parser and manual analytic methods can be employed for log analysis.

Network Traffic Analysis: Network traffic analysis involves monitoring the traffic between systems to detect any unusual activity related to API-based attacks. This technique enables the identification of the origin and destination of API calls and the type of data being transferred. Specialized tools like Wireshark are commonly used for network traffic analysis.

System Profiling: System profiling entails gathering information about the system's configuration to identify potential vulnerabilities or weaknesses. System profiling aims to pinpoint elements susceptible to API-based attacks by scrutinising software and hardware configurations. Automated tools such as OSSEC or manual analysis techniques can be employed for system profiling.

Evidence Collection: Evidence collection encompasses properly gathering and preserving evidence associated with API-based attacks. This includes collecting system logs, memory dumps, and network traffic data. Ensuring the meticulous collection of evidence is essential to ensure its admissibility in court and its ability to support potential legal actions if required.

In API forensics, employing these techniques facilitates the systematic retrieval, analysis, and preservation of evidence, investigating API-based attacks and supporting potential legal proceedings.

13. API FORENSICS TOOLS

There are various tools available for conducting API forensics. Here is an expanded version of the provided information:

Volatility Framework: The Volatility Framework is an open-source memory forensics tool widely used for API forensics. It enables detecting of malicious behaviour associated with API-based attacks by analyzing system memory dumps. With Volatility, investigators can examine the memory artefacts to identify signs of compromise or the presence of malware targeting APIs.

Wireshark: Wireshark is a popular network traffic analysis tool used for monitoring and analyzing network traffic in API-based attacks. It captures and decodes network traffic in real-time, allowing analysts to study the data transferred during API calls. Wireshark facilitates identifying any suspicious or malicious activities, enabling detailed analysis of network communication related to APIs.

Log Parser: Log Parser is a versatile log analysis tool for parsing and analyzing system logs relevant to API-based attacks. It extracts pertinent information from log files, aiding in understanding API call behaviour and identifying any anomalies or malicious activities. Log Parser is valuable in extracting insights from system logs and facilitating investigation.

OSSEC: OSSEC is a host-based intrusion detection system commonly used for system profiling and the detection and prevention of API-based attacks. It monitors various aspects, such as system logs, file changes, and network traffic, to identify suspicious activity. OSSEC generates alerts to notify administrators when potential API-related threats or anomalies are detected, contributing to enhanced security and incident response.

Fiddler: Fiddler is a web debugging proxy tool widely employed for analyzing and debugging HTTP traffic associated with API-based attacks. It captures and analyzes HTTP traffic between clients and servers, allowing investigators to identify and investigate malicious activities within the API calls. Fiddler provides insights into the communication between clients and APIs, aiding in identifying potential security issues or vulnerabilities.

These tools serve as valuable assets for conducting effective API forensics, providing memory analysis, network traffic monitoring, log analysis, system profiling, and HTTP traffic inspection capabilities. Leveraging these tools can significantly enhance the investigation process and aid in identifying and responding to API-based security incidents.

14. DISCUSSION AND FUTURE STUDY

API malware analysis and forensics research can explore several potential directions to enhance detection and response capabilities. These areas of study include:

Advancement of Machine Learning-Based Techniques: Further development of machine learning-based techniques can lead to more accurate detection and classification of API-based attacks. By training models on large datasets of known attack patterns, researchers can enhance the ability to identify and categorize malicious API activities with high precision.

Real-Time Detection and Response Systems: There is a need for automated systems that can detect API-based attacks in real-time and respond promptly and effectively. Developing intelligent algorithms and frameworks capable of monitoring API traffic and identifying suspicious behaviours in real-time can significantly improve incident response and mitigate the impact of attacks.

Analyzing Encrypted Traffic: As encryption becomes more prevalent, developing techniques to analyze encrypted traffic is crucial for detecting and mitigating API-based attacks. Exploring methods for identifying suspicious activities within encrypted API communications can help uncover malicious intentions and potential security breaches.

Addressing API Security in Emerging Technologies: The emergence of technologies like cloud computing and the Internet of Things (IoT) presents new challenges for API security. Future research should focus on developing robust security solutions tailored to these technologies, ensuring that APIs used in cloud-based systems and IoT applications are adequately protected.

In terms of implications for software development and cybersecurity best practices, prioritizing API security is paramount. Developers should receive training in secure coding practices and adhere to established security guidelines. Regular security assessments and testing should be conducted to identify and address vulnerabilities in API-based applications, helping to strengthen the overall security posture.

The industry can proactively address emerging threats and protect systems and data from API-based attacks by pursuing these future research directions and emphasising API security in software development.

15. CONCLUSION

The paper provides a comprehensive examination of API malware analysis and forensics. It covers various aspects, including the role of APIs in software development, the risks associated with API malware attacks, and the significance of API malware analysis and forensics. Additionally, it delves into common types of API malware attacks, techniques for detecting and analyzing API malware, methods for analyzing API calls to identify malicious code, and tools utilized in API forensics.

The study underscores the criticality of API security and emphasizes the importance of conducting regular security assessments and testing to identify vulnerabilities in API-based applications. It stresses the need for developers to adhere to established security guidelines, receive training in secure coding practices, and establish incident response plans to address API-based attacks effectively.

The paper further advocates for future research in several key areas. It suggests the development of automated systems capable of real-time detection of API-based attacks, techniques for analyzing encrypted traffic associated with APIs, and security solutions tailored to emerging technologies like cloud computing and the Internet of Things (IoT). Furthermore, it highlights the potential of machine learning-based techniques in API malware analysis and forensics. The research should also focus on evaluating the effectiveness of different API malware detection and analysis techniques and establishing best practices for API security. Overall, the paper emphasizes the significance of API malware analysis and forensics, provides insights into effective security measures, and outlines future research directions to enhance API security and combat emerging threats.

REFERENCES

[1] Kim, D., Kim, T. H., & Yeom, K. (2020). A Survey on Machine Learning-Based Malware Detection Using API Call Sequences. Journal of Information Processing Systems, 16(6), 1422-1435.

[2] Rajesh, R., & Karthick, S. (2020). Malware detection using API call sequence analysis with deep

learning techniques. Cluster Computing, 23(2), 1103-1116.

[3] Chowdhury, S. R., Samanta, S., & Roy, D. (2020). Malware detection using API call sequences: A comparative study. In 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC) (pp. 104-109). IEEE.

[4] Dhiman, G., & Juneja, D. (2019). Malware detection using API calls and machine learning. International Journal of Network Security, 21(1), 68-75.

[5] Khalid, H., Rasool, R., & Mehmood, Z. (2018). Malware detection using API call sequence and Deep Learning.
In 2018 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST) (pp. 21-26). IEEE.

[6] Youn, C. H., Seo, S. H., & Kim, S. J. (2018). Deep learning-based malware detection using API call sequences. In 2018 20th Asia-Pacific Network Operations and Management Symposium (APNOMS) (pp. 1-4). IEEE.

[7] Santos, I., Rocha, G., & de Carvalho, A. (2014). A dynamic feature approach to malware detection using API calls. In International Symposium on Research in Attacks, Intrusions, and Defenses (pp. 119-139). Springer.

[8] Wang, Y., Li, H., & Yu, C. (2017). Malware detection using API call sequences. Journal of Ambient Intelligence and Humanized Computing, 8(6), 1117-1124.

[9] Kim, J., Kang, B. J., & Kim, H. (2016). A study on API-based malware detection using machine learning techniques. Journal of Information Processing Systems, 12(1), 66-80.

[10] Seo, S. H., Kim, D. J., & Kim, S. J. (2016). A malware detection method using API call sequence and reinforcement learning. In International Symposium on Ubiquitous Networking (pp. 343-348). Springer.

[11] Qiao, Y., Yang, Y., He, J., Tang, C., & Liu, Z. (2014). CbmCBM: free, automatic malware analysis framework using API call sequences. In Knowledge Engineering and Management.

[12] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys (CSUR), 41(3), 1-58.
[13] Acosta, J. C., Mendoza, H., & Medina, B. G. (2012). An efficient common substrings algorithm for on-the-fly behavior-based malware detection and analysis. In Proceedings - IEEE Military Communications Conference MILCOM.

[14] Ki, Y., Kim, E., & Kim, H. K. (2015). A Novel Approach to Detect Malware Based on API Call Sequence Analysis. International Journal of Distributed Sensor Networks.

[15] Sundarkumar, G. G., & Ravi, V. (2013, December). Malware detection by text and data mining. In 2013 IEEE International Conference on Computational Intelligence and Computing Research.

[16] Yu, H., Sun, X., Chen, S., Tang, F., & Chen, Z. (2018). Deep learning-based API-call sequence embedding model for malware detection. Future Generation Computer Systems, 86, 431-440.

[17] Chen, T., Shu, J., Huang, T., Wang, Y., & Xu, G. (2020). Malware detection using API call sequences with recurrent neural networks. IEEE Access, 8, 146029-146039.

[18] Abbas, H., Dsouza, M., Alazab, M., & Venkatraman, S. (2018). Malware detection using deep learning techniques based on API call sequences. In Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (pp. 1282-1289). IEEE.

[19] Tang, S., Cao, Y., Huang, H., & Wang, X. (2019). A malware detection method based on API call sequences and machine learning. Security and Communication Networks, 2019.

[20] Wang, Q., Liu, Z., & Xie, T. (2020). Malware detection using API call sequence and CNN-based

feature extraction. Journal of Ambient Intelligence and Humanized Computing, 11(8), 3353-3363.

[21] Liu, Z., Wang, Q., Xie, T., & Li, Y. (2020). Malware detection using LSTM-based API call sequence feature extraction. IEEE Access, 8, 160157-160167.

[22] Liao, Y., Liu, X., & Zhang, Y. (2021). Malware detection using ensemble learning with deep neural networks based on API call sequences. IEEE Access, 9, 72091-72100.