

# Cognitive routing in software defined networks using learning models with latency and throughput constraints

Nagaraju Tumakuru Anadanaiah<sup>1,2</sup>, Malode Vishwanatha Panduranga Rao<sup>3</sup>

<sup>1</sup>Electronics Engineering, Faculty of Engineering and Technology, JAIN (Deemed-to-be University), Bengaluru, India

<sup>2</sup>Department of Electronics and Communication Engineering, Government Engineering College, Ramanagara, India

<sup>3</sup>Computer Science & Engineering, Faculty of Engineering & Technology, JAIN (Deemed-to-be University), Bengaluru, India

## Article Info

### Article history:

Received Sep 20, 2023

Revised Oct 8, 2023

Accepted Oct 21, 2023

### Keywords:

Cognitive routing

Cognitive routing engine

Reinforcement learning

Round trip time

Software defined networks

## ABSTRACT

To address latency and throughput challenges in software defined networks (SDNs), the research investigates cognitive routing's revolutionary implications. In today's data-driven world, network performance optimisation is crucial. Cognitive routing is a dynamic and potentially disruptive network management technology. Cognitive routing, strengthened by reinforcement learning and adaptive decision-making, is crucial to network efficiency and responsiveness, according to our study. The results show that cognitive routing optimises performance by limiting delay and maximising throughput. SDN application cognitive routing engine (CRE) driving forces, design, and preliminary assessment are described in this article. The CRE finds almost optimal paths for a user's quality of service (QoS) need while minimising monitoring overhead. Instead of global monitoring to find optimal paths, local monitoring achieves this. In ad-hoc networks, finding a trustworthy path reduces latency and ensures network stability. The proposed system was simulated utilising many parameters. Compared to previous SDN-based systems, end-to-end latency and ping round-trip time were better.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



## Corresponding Author:

Nagaraju Tumakuru Anadanaiah

Department of Electronics & Communication Engineering, Government Engineering College

Ramanagara, Karnataka, India

Email: nagarajuta76@gmail.com

## 1. INTRODUCTION

In the task of managing and running computer networks, the introduction of software defined networks (SDNs) marks a significant shift in strategy. SDN is a design approach that isolates the control plane (which decides on routing) from the data plane (which sends packets) [1]. Because of this split, network managers may more easily monitor and control network resources in a centralised, programmable fashion, which has several advantages. With SDNs, the duties of controlling the network are separated into a single, logical component called the controller or SDN controller. This consolidation offers a bird's-eye perspective of the network, letting admins make informed choices depending on the status of the system and the volume of traffic. SDNs increase a network's adaptability and flexibility. Software applications can specify and modify network policies and configurations by communicating with the SDN controller through well-defined application programming interfaces (APIs) [2]. The capacity to modify and roll out new services is simplified by this programmability.

With SDNs, traffic can be engineered and optimised at a granular level. Network performance, congestion, and resource utilisation can all be enhanced by SDNs' ability to dynamically reroute traffic depending on real-time conditions and requirements. The scalability of SDNs is built in. The SDN controller can effectively manage the network even when the number of connected devices develops in tandem with the network's traffic load [3]. SDNs

allow for more effective distribution of network assets. SDNs can prevent wasteful over-provisioning of network resources by the dynamic modification of network pathways and optimisation of traffic flow.

Applications that depend on real-time responsiveness and quick data transfer necessitate networks with low latency and high throughput. However, there are several obstacles that must be overcome in SDNs before low latency and high throughput can be achieved concurrently. The first is network congestion management. Congestion in a network happens when a certain connection or portion of the network is overloaded with data. The result may be slowed connections and longer wait times [4]. Congestion management techniques, used by SDNs to identify and relieve congestion in real time by rerouting traffic to less crowded pathways or modifying quality of service (QoS) policies on the fly, are a necessary component of any successful SDN.

The second is QoS enforcement. To guarantee low latency and high throughput, QoS standards must be strictly enforced to give some traffic types more priority than others. It can be difficult to strike a balance between these two goals. For SDNs to function, QoS mechanisms must be implemented to distribute network resources in accordance with specified criteria, giving priority to latency-sensitive traffic without depriving other traffic types of enough bandwidth [5].

The third is path selection and optimization. It might be difficult to figure out where data packets should go in a network if they need to achieve both low latency and high throughput. These considerations may not be considered sufficiently by common routing systems. To dynamically route traffic for best performance, SDNs require the use of sophisticated path selection algorithms that take into consideration variables such as latency, available bandwidth, and network load [6].

The fourth is latency-sensitive applications. Real-time video conferencing and online gaming are two examples of applications that are very sensitive to latency. It's possible that these programmes can't find the low-latency links they need using conventional routing [7]. SDNs may prioritise latency-sensitive traffic by determining which applications are most affected by delays and allocating resources accordingly.

The fifth is network slicing for diverse traffic types. Different types of traffic need different network characteristics, such as latency and throughput. It might be difficult to guarantee top performance for every sort of traffic. Network slicing is a feature of SDNs that enables the construction of separate, optimised network segments for various types of traffic. You may adjust the latency and throughput of each slice separately [8]. SDNs require careful consideration of network congestion, QoS policies, dynamic traffic patterns, path optimisation, application sensitivity, hardware capabilities, security, and the capacity to manage varied traffic types if low latency and high throughput routing are to be achieved. SDNs offer the adaptability and programmability essential for overcoming these obstacles and maximising network performance.

Using cutting-edge technology like machine learning (ML) and artificial intelligence (AI), cognitive routing takes an intelligent and adaptable approach to network routing to improve routing decisions and overall network performance [9]. It surpasses conventional routing algorithms by constantly analysing network data and making real-time adjustments to routing choices to accommodate user-defined requirements. Cognitive routing is a form of intelligent routing that mainly utilises machine learning algorithms and other forms of artificial intelligence [10]. These algorithms consider the status of the network, past traffic patterns, and other factors to determine the most efficient routes to take. Cognitive routing relies heavily on its ability to learn and adapt over time. With time and experience, the system learns to make better routing decisions. It can quickly adjust to new traffic patterns and evolving network circumstances because of its flexibility. Decisions made by cognitive routing systems can be dynamically modified to meet user-defined goals and limitations. Low latency, high throughput, and QoS needs may all be met because of this flexibility [11].

In the context of today's networked world, the suggested study on "Cognitive routing in software defined networks using learning models with latency and throughput constraints" is of great importance. Optimising network performance is essential due to the ever-increasing need for low-latency, high-throughput network services. Cognitive routing's flexibility to respond to changing conditions and limits holds great promise for enhancing both network performance and user satisfaction. Intelligent routing solutions are more important than ever with the rise of 5G networks, edge computing, the internet of things (IoT), and data-intensive applications. Cognitive routing can assist networks in adjusting to the peculiar demands and difficulties of such innovations. As network infrastructures grow, scalable routing solutions become increasingly important. Because of its flexibility and capacity to learn, cognitive routing is suitable for use in networks of varying sizes. The potential for the proposed works to advance network technology and enhance the overall quality of network services stems from their applicability across a wide range of industries.

## 2. COGNITIVE ROUTING IN SOFTWARE DEFINED NETWORKS

Most existing approaches to SDN traffic engineering work on the assumption that, in a logically centralised controller, SDN provides an overview of the network's current status and topology, allowing network optimisation via various global traffic engineering techniques such as constrained shortest path first. In this research, we claim it is inefficient to collect complete and up-to-date status data for the entire network

at the frequency needed for optimal traffic optimisation because of how vast networks might be. The present study utilises the cognitive routing engine (CRE) within the context of cognitive packet networks (CPNs) [12], [13]. In this framework, individual routers employ distinct routing and learning algorithms for each source-destination pair and QoS specification. Additionally, the network status is gathered through the utilisation of intelligent packets and acknowledgement packets. As opposed to CRE, which operates decentralized on top of the SDN controller and relies solely on OpenFlow (OF) methods to ascertain network status, OF is a conceptually centralised layer that handles all network state gathering [14]–[16]. Figure 1 illustrates the integration of the CRE application inside the SDN architecture and its interaction with various components in a standard deployment scenario.

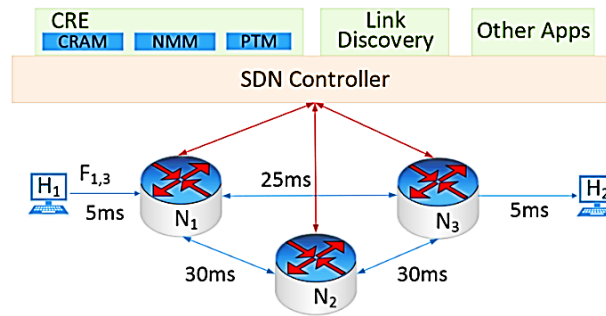


Figure 1. Network architecture

The steps in message exchange are:

- The network forwarding element (NFE) N1 receives a fresh flow F1,3 that must reach the NFE N3.
- The network function element N1 is unable to locate a corresponding rule inside its internal flow tables for the initial packet of flow F1,3. N1 encapsulates the packet within an OF Packet-In message and sends it to the master controller. This action is triggered by the NFE's default behaviour in response to a table-miss event, which is to forward the packet.
- To enhance the interpretability of the OF Packet-In message for the many applications that function on top of the SDN controller, it will be subjected to dissection.
- The data format delivered by the controller allows the apps that were notified to determine whether they wish to take any action.
- After the Link Discovery programme has uncovered the topology of the network, CRE will install a route for flow F1,3 by determining the shortest route between NFE N1 and N2. The network monitoring module (NMM)'s job in CRE is to collect data on the network's status so that cognitive routing algorithm module (CRAM) can function.
- For the CRAM to compute the numerical value of the objective function of every route it has chosen, it sends requests to the NMM for specific characteristics of connections in the network and the NMM responds by updating the recurrent neural networks (RNNs) with return loss (RL) so that the CRAM can select more suitable routes in future attempts. Because of its intelligence, NMM doesn't need to constantly poll the network for fresh information; instead, it uses its cache of previously collected data to respond to CRAM's queries.
- After receiving input from the NMM, CRAM uses RL to update its RNNs and then outputs the optimal routes. The "Path-to-of translator module" (PTM) is then given the routes to update if necessary.
- The PTM must alter the flow table(s) inside national fire equipment (NFEs) to implement the routes such that it prevents packet loss and forwarding loops.

## 2.1. Reinforcement learning in SDNs

The concepts at the heart of ML are not very complicated; they are essentially a model of how humans learn and develop. To put it another way, the ML approach relies more on induction and synthesis than deduction to arrive at its conclusions as shown in Figure 2. Different criteria may be used to categorise ML approaches. The ML models may be broken down into several categories depending on the type of task being performed, including regression, classification, and structured learning. Many classification and prediction issues have been successfully solved using ML algorithms [17]. Here we look at ML algorithms from a classification standpoint and demonstrate a traditional ML approach used in SDNs.

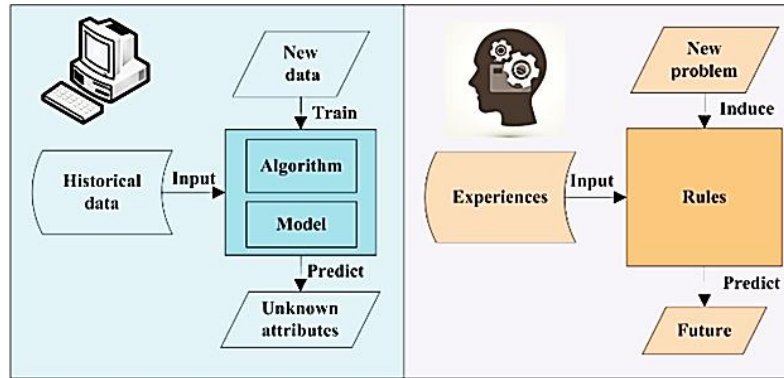


Figure 2. A comparison of ML reasoning with that of humans

An agent learns to reward guided behaviour through trial and error using RL, in which the agent receives reinforcement from its interactions with the environment. To maximise the reinforcement signal, an RL system (RLS) would constantly fine-tune its settings. The environment's reinforcement signal is an appraisal of the outcome, good or poor, rather than instruction on how to generate the desired behaviour. According to our findings, RL is typically employed to promote robustness and scalability [18], [19], and it enables choosing routes or route optimisation in SDNs [20], [21]. When delay minimise and throughput enhancement are used as the operation and maintenance approach for deterministic policy gradient (DDPG) routing optimization mechanism (DROM) [22], the network's performance is enhanced with reliable and superior routing services, and convergence and effectiveness are boosted. By detecting and learning from the internet of vehicles (IoV) environment, software defined cognitive routing (SDCoR) [23] is the first research to present an optimum routing strategy that outperforms multiple standard IoV protocols. Having bigger packet bucket sizes and reducing the number of consecutive packets traversing various pathways are both effective solutions for dealing with the primary problem posed by a high level of jitter [24].

Some cutting-edge RL study proposals propose combining RL with additional technologies for enhanced performance. For instance, RNNS with RL is designed to discover the best overlay pathways with little maintenance [25]. Research into auto-scaling policy decisions led researchers to SRSA [26], a decision mechanism based on RL. In addition, we explore RL with architectural modifications due to the complexity and variability of the network environment. To effectively manage networks enabled by Self-Organizing Networks, [27] presented a scalable solution based on distributed RL. Combining DL with RL, deep reinforcement learning (DRL) boosts the efficiency and accuracy of RL programmes. DRL has made tremendous theoretical and practical advances.

### 3. COGNITIVE ROUTING ALGORITHM

Routing methods for mobile ad hoc networks have proliferated in recent years. To carry out routing tasks, they rely mostly on real-time rather than forecasted factors. They have no idea the parameters have changed over time. The channel conditions and link load are ignored by most conventional routing systems. In this scenario, it is presumed that all links experience the identical channel conditions and load levels.

In this approach, RNN and RL are the basis of CRAM, a routing method. The shortest path for a particular flow is then used to train a new RNN at each NFE along that path. Each neuron in an RNN is like an open port on an NFE, and the entire network works together to make predictions. The RNN might use an experimental or exploitative approach when deciding which output to utilise for the next layer. In exploration mode, the RNN picks an output port at random; in exploitation method, it picks the neuron with the greatest probability. In the absence of an existing RNN for this flow at the next hop NFE, a new RNN is trained to represent it.

During the exploitation phase of the RNN, the selection of the next node in the NFE process is influenced by the individual excitation levels of each neuron. To determine neuron  $i$ 's potential ( $q_i$ ), we can use (1).

$$q_i = \frac{\lambda_i^+}{r_i + \lambda_i^-} \quad (1)$$

The collective positive and negative potentials originating from all interconnected neurons in the RNN and affecting neuron  $i$  are represented as  $\lambda^+$  and  $\lambda^-$ , respectively. These global potentials may be estimated as follows, considering both the individual neuronal potentials and the connection weights,

$$\lambda_i^+ = \sum_{j \in N} q_j w_{j,i}^+ + \Lambda_i^+ \text{ where } j \neq i \quad (2)$$

$$\lambda_i^- = \sum_{j \in N} q_j w_{j,i}^- + \Lambda_i^- \text{ where } j \neq i \quad (3)$$

The constant rates at which external positive and negative pulses reach a neuron are denoted by  $\Lambda_i^+$  and  $\Lambda_i^-$ . The normalisation factor  $r_i$  is determined by (4):

$$r_i = \sum_{j \in N} [w_{i,j}^+ + w_{i,j}^-] \text{ where } j \neq i \quad (4)$$

Each link's positive  $w_{i,j}^+$  weight and negative  $w_{i,j}^-$  weight is calculated via RL. The following expression determines the exponential average value of the objective function  $O_{sd}[t]$  of an entire route.

$$O_{sd}[t] = \alpha O_{sd}[t-1] + (1-\alpha)(o_{sd}[t]) \text{ where } 0 < \alpha < 1 \quad (5)$$

Where  $O_{sd}[t-1]$  is the objective function's value at time step t1,  $O_{sd}[t]$  is the objective function's value at time step t and  $\alpha$  is an exponential average parameter that defines how much weight t1 should be given to t1's value in the objective function. The link weights  $W$  are renormalized such that they don't keep growing forever by first determining the revised  $r_i$  according to the revised  $W$  values in (3). Furthermore, CRAM effectively preserves a record of the most recently identified pathways for each route query. If the current route has been in operation for a duration beyond the minimal threshold, the optimal path from the available options is employed for the actual transmission of traffic.

### 3.1. Latency monitoring

The collection of connection latency information is a crucial aspect of our approach. We adopted the method described in [28], which generates fabricated Ethernet packets of type 0 to 07c3. The payload of the transmitted data includes the data path ID and timestamp from the transmitting device. To adhere to the prescribed Network packet size of 64 bytes, the packets undergo a process known as "padding," wherein zeros are added to the packets. Every second, the packets are transmitted to all nearby switches. The advantage of this method is that the measured latency coincides with the latency experienced by the flows.

We are aware that packet reordering may occur during the process of changing from one routing configuration to another. As a result, it is preferable to reduce the frequency with which routing configurations are altered. Given that a learning strategy necessitates making routing configuration updates, it is desirable to minimise the time it takes for the network to converge. Most of the convergence traffic remains unrouted. Therefore, the decision is between encountering significant traffic congestion, experiencing extended delays and lengthy processing times, or opting for frequent adjustments in routing with shorter processing times and occasional packet disarray. Congestion, which increases latency and decreases throughput, may be detected by measuring the network's latency, making it a helpful indication of the network's performance.

## 4. PERFORMANCE EVALUATION

Using the SDN emulator Mininet and NS-3, we tested the suggested method. With this setup, nodes were moving in three distinct groups, each with a 100-meter communication range in a 2000-by-2000-meter target area. All the nodes in a cluster travelled at the same rate. However, the rates at which the individual clusters travelled ranged from around 0 to 25 m/s. The spectrum band was split into  $M = 5$  channels, with each channel being able to house one of two licenced primary users (PUs) and their respective 500 m communication ranges. These land-based PU nodes were modelled as having an on/off activity pattern exponential in shape, with a rate parameter of 0.05. End-to-end delay performance using NS-3 is depicted in Figure 3 as a function of the total number of nodes and various PU idleness probabilities. The end-to-end latency is the total amount of time it takes for a packet to travel from its initial source node to its destination node, including all intermediate hops. As the number of connected devices rises in the testing environment, the average network latency reduces. The latency lowers across the board as network density and the chance of the PU being in an idle state are both raised. This is because of the enhanced connectedness brought about by higher network densities. In contrast, the second scenario results in more available channels due to the PU's increased likelihood of being in an idle state. All the suggested and reference systems have a large end-to-end delay when the network density is low, meaning the network is sparse. This is because the requesting node experiences a significant delay since it is unable to locate any other relay nodes with which to establish a stable connection. Cluster heads (CHs) in our proposed approach, however, make it possible for a querying node to establish a reliable connection with a gateway that acts as a relay node, even in sparsely populated networks. Therefore, the suggested scheme's CH selection further reduces the time.

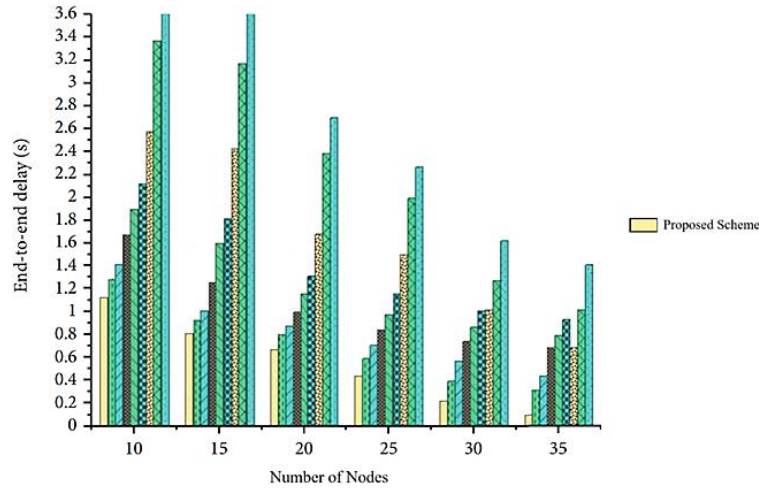


Figure 3. Performance comparison with modified settings

Mininet is an open-source SDN emulator that allows researchers and developers to create virtual SDN networks for testing and experimentation. It provides a platform for emulating network topologies, hosts, switches, and controllers, enabling users to simulate complex network environments without the need for physical hardware. Mininet is employed for the purpose of detecting delays and facilitating route switching within the network shown. Using the topology shown in Figure 1, this simulation demonstrates how CRE may locate, monitor, and switch pathways in a network where the latency on links might fluctuate. Upon entering the network, Ping establishes a connection between source 1 and destination 2. The CRE proceeds to install two paths:  $P_{1,2} = H1 \rightarrow N1 \rightarrow N2 \rightarrow H2$  and  $P_{2,1} = H2 \rightarrow N2 \rightarrow N1 \rightarrow H1$ . The routes that exhibit the minimum hop count also provide the shortest RTT of 130 ms. The initial Ping has a greater RTT compared to subsequent Pings. The act of informing the controller about newly generated flows, calculating optimal routes, and implementing regulations inside NFEs results in additional operational burden. Subsequently, the CRE commences network monitoring utilising RNN in conjunction with RL. At the 10-second mark of the trial, there was a noticeable increase in delay for the connections connecting N1 to N2, namely L1,2 and L2,1. The delay rose from 20ms to 200ms, thus causing the Ping RTT to rise to 430 ms. The rise in path delays is identified by the CRE system, which monitors the network. The initial path that undergoes modification is P2,1. An alternate path,  $P_{2,1} = H2 \rightarrow N2 \rightarrow N3 \rightarrow N1 \rightarrow H1$ , is discovered, resulting in a reduction of the RTT to 260 ms at the 18th Ping iteration.

Additionally, the path denoted as P1,2 is modified to new P1,2, which represents the route  $H1 \rightarrow N1 \rightarrow N3 \rightarrow N2 \rightarrow H2$ . This alteration results in the RTT to 145 ms by the 20th ping as shown in Figure 4. The two recently discovered pathways identified by CRE have been determined to be the most optimum options for the new network circumstances.

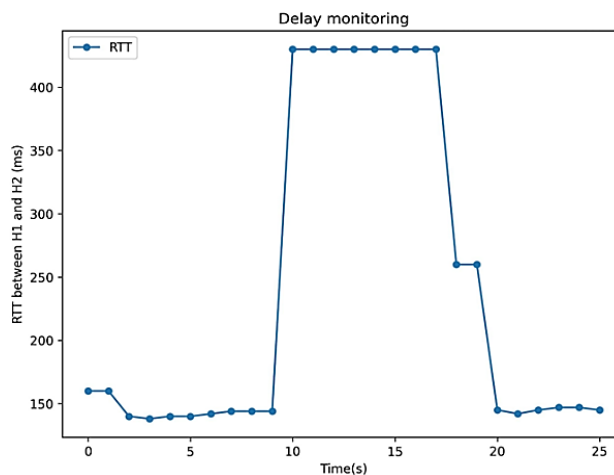


Figure 4. Delay monitoring and path switching

According to the findings presented in Table 1, it is evident that the utilisation of CRE can significantly decrease the amount of monitoring necessary to identify an optimal path. Specifically, the reduction can reach up to 8.6 times the original requirement. Moreover, the average deviation from the optimal RTT when using CRE is only 2.0%. However, it is worth noting that the process of identifying the best path with CRE does require additional time, approximately 37.63 seconds more, when compared to the standard approach. This time difference is based on a minimum update frequency of 5 seconds.

Table 1. Analysis of CRE ping RTTs

Experiment iterations	Amount of optimal monitoring	Number of CRE monitoring	CRE time (s)	Increased CRE ping RTTs (%)
1	350	30	33.5	3.4
2	350	60	38.6	0.4
3	290	40	43.4	2.0
4	350	25	35	2.2
Average	335	38.75	37.63	2.0

## 5. CONCLUSION

This research effort has explored the domain of cognitive routing in SDNs, providing insight into its crucial function in tackling the significant obstacles posed by latency and throughput limitations. The study conducted emphasises the revolutionary nature of cognitive routing, which leverages machine learning and adaptive decision-making to greatly improve network performance. The primary results underscore the significance of cognitive routing in enhancing network performance while concurrently achieving a harmonic equilibrium between the sometimes-opposing objectives of minimising latency and maximising throughput. The system's ability to respond in real-time guarantees that there are minimum delays in transmitting data packets, making it well-suited for applications that need low latency. Additionally, it maximises network efficiency for operations that involve large amounts of data.

In this study, we demonstrate that the CRE algorithm can identify pathways that closely approximate the best solutions, while minimising the additional monitoring costs. The performance may be achieved by CRE without necessitating any alterations to the pre-existing SDNs that operate on OpenFlow. The management of the network is overseen by the primary controller, while the local controllers, known as CHs, reduce the quantity of control messages and network latency. In the end, it is important to emphasise once again that the incorporation of learning models into cognitive routing algorithms has great potential for enhancing network performance on an ongoing basis. The use of these models allows for cognitive routing to emerge as a transformative force in the quest for efficient, responsive, and dependable networks in the modern day. This is made possible by the predictive abilities, ongoing learning, and tailored prioritisation facilitated by these models. There are several limitations that arise while using the SDN strategy. For the protocol to function properly, it is imperative that the centralised controller remains in its active state. Failure to do so would result in the protocol relying solely on localised views of the network, which would provide challenges in identifying alternative pathways. The performance of a system is contingent upon the connection of several nodes. This study serves as a strong appeal to both the academic and industrial sectors to go deeper into the investigation and use of cognitive routing to fully harness its capabilities in improving network operations and user experiences.

## REFERENCES





- [1] D. M. Casas-Velasco, O. M. C. Rendon, and N. L. S. Da Fonseca, "DRSIR: A deep reinforcement learning approach for routing in software-defined networking," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4807–4820, Dec. 2022, doi: 10.1109/TNSM.2021.3132491.
- [2] Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang, and Y. Sun, "A survey of networking applications applying the software defined networking concept based on machine learning," *IEEE Access*, vol. 7, pp. 95397–95417, 2019, doi: 10.1109/ACCESS.2019.2928564.
- [3] A. V. Kordali and P. G. Cottis, "A contract-based spectrum trading scheme for cognitive radio networks enabling hybrid access," *IEEE Access*, vol. 3, pp. 1531–1540, 2015, doi: 10.1109/ACCESS.2015.2455492.
- [4] Y. Tianfang and Q. Xuesong, "STCC: A SDN-oriented TCP congestion control mechanism for datacenter network," *IET Networks*, vol. 10, no. 1, pp. 13–23, Dec. 2021, doi: 10.1049/ntw2.12005.
- [5] M. Karakus and A. Durrresi, "Quality of service (QoS) in software defined networking (SDN): A survey," *Journal of Network and Computer Applications*, vol. 80, pp. 200–218, Feb. 2017, doi: 10.1016/j.jnca.2016.12.019.
- [6] B. Isyaku, K. A. Bakar, M. S. M. Zahid, E. H. Alkhamash, F. Saeed, and F. A. Ghaleb, "Route path selection optimization scheme based link quality estimation and critical switch awareness for software defined networks," *Applied Sciences (Switzerland)*, vol. 11, no. 19, p. 9100, Sep. 2021, doi: 10.3390/app11199100.
- [7] E. Lakiotakis, C. Liaskos, and X. Dimitropoulos, "Application-network collaboration using SDN for ultra-low delay teleorchestras," in *Proceedings - IEEE Symposium on Computers and Communications*, Jul. 2017, pp. 70–75, doi: 10.1109/ISCC.2017.8024507.
- [8] D. Alotaibi, V. Thayanathan, and J. Yazdani, "The 5G network slicing using SDN based technology for managing network traffic," *Procedia Computer Science*, vol. 194, pp. 114–121, 2021, doi: 10.1016/j.procs.2021.10.064.
- [9] D. M. Casas-Velasco, O. M. C. Rendon, and N. L. S. Da Fonseca, "Intelligent routing based on reinforcement learning for software-







- defined networking,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 870–881, Mar. 2021, doi: 10.1109/TNSM.2020.3036911.
- [10] H. Ghafoor and I. Koo, “Cognitive routing in software-defined underwater acoustic networks,” *Applied Sciences (Switzerland)*, vol. 7, no. 12, p. 1312, Dec. 2017, doi: 10.3390/app7121312.
- [11] Y. J. Wu, P. C. Hwang, W. S. Hwang, and M. H. Cheng, “Artificial intelligence enabled routing in software defined networking,” *Applied Sciences (Switzerland)*, vol. 10, no. 18, p. 6564, Sep. 2020, doi: 10.3390/APP10186564.
- [12] P. P. Ray, “A survey on cognitive packet networks: Taxonomy, state-of-the-art, recurrent neural networks, and QoS metrics,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, pp. 5663–5683, Sep. 2022, doi: 10.1016/j.jksuci.2021.05.017.
- [13] W. Serrano and E. Gelenbe, “Deep learning clusters in the cognitive packet network,” *Neurocomputing*, vol. 396, pp. 406–428, Jul. 2020, doi: 10.1016/j.neucom.2018.07.101.
- [14] R. Wazirali, R. Ahmad, and S. Alhiyari, “Sdn-openflow topology discovery: An overview of performance issues,” *Applied Sciences (Switzerland)*, vol. 11, no. 15, p. 6999, Jul. 2021, doi: 10.3390/app11156999.
- [15] W. Li, W. Meng, and L. F. Kwok, “A survey on OpenFlow-based software defined networks: security challenges and countermeasures,” *Journal of Network and Computer Applications*, vol. 68, pp. 126–139, Jun. 2016, doi: 10.1016/j.jnca.2016.04.011.
- [16] A. Vishnu Priya and N. Radhika, “Performance comparison of SDN OpenFlow controllers,” *International Journal of Computer Aided Engineering and Technology*, vol. 11, no. 4/5, p. 467, 2019, doi: 10.1504/ijcaet.2019.10020284.
- [17] S. Prabhakaran *et al.*, “Predicting attack pattern via machine learning by exploiting stateful firewall as virtual network function in an SDN network,” *Sensors*, vol. 22, no. 3, p. 709, Jan. 2022, doi: 10.3390/s22030709.
- [18] S. Lu, J. Wu, J. Shi, P. Lu, J. Fang, and H. Liu, “A dynamic service placement based on deep reinforcement learning in mobile edge computing,” *Network*, vol. 2, no. 1, pp. 106–122, Feb. 2022, doi: 10.3390/network2010008.
- [19] S. Zehra *et al.*, “Machine learning-based anomaly detection in NFV: A comprehensive survey,” *Sensors*, vol. 23, no. 11, p. 5340, Jun. 2023, doi: 10.3390/s23115340.
- [20] C. Yu, J. Lan, Z. Guo, and Y. Hu, “DROM: Optimizing the Routing in Software-Defined Networks with Deep Reinforcement Learning,” *IEEE Access*, vol. 6, pp. 64533–64539, 2018, doi: 10.1109/ACCESS.2018.2877686.
- [21] T. V. T. Duong and L. H. Binh, “IRSML: An intelligent routing algorithm based on machine learning in software defined wireless networking,” *ETRI Journal*, vol. 44, no. 5, pp. 733–745, Aug. 2022, doi: 10.4218/etrij.2021-0212.
- [22] C. Wang, L. Zhang, Z. Li, and C. Jiang, “SDCoR: Software defined cognitive routing for internet of vehicles,” *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3513–3520, Oct. 2018, doi: 10.1109/JIOT.2018.2812210.
- [23] M. D. S. Islam, M. Al-Mukhtar, M. D. R. K. Khan, and M. Hossain, “A survey on SDN and SDCN traffic measurement: Existing approaches and research challenges,” *Eng*, vol. 4, no. 2, pp. 1071–1115, Apr. 2023, doi: 10.3390/eng4020063.
- [24] Imran, Z. Ghaffar, A. Alshahrani, M. Fayaz, A. M. Alghamdi, and J. Gwak, “A topical review on machine learning, software defined networking, internet of things applications: Research limitations and challenges,” *Electronics (Switzerland)*, vol. 10, no. 8, p. 880, Apr. 2021, doi: 10.3390/electronics10080880.
- [25] Q. M. Salih *et al.*, “Dynamic channel estimation-aware routing protocol in mobile cognitive radio networks for smart IIoT applications,” *Digital Communications and Networks*, vol. 9, no. 2, pp. 367–382, Apr. 2023, doi: 10.1016/j.dcan.2023.01.019.
- [26] A. Llorens-Carrodegua, I. Leyva-Pupo, C. Cervelló-Pastor, L. Piñeiro, and S. Siddiqui, “An SDN-based solution for horizontal auto-scaling and load balancing of transparent VNF clusters,” *Sensors*, vol. 21, no. 24, p. 8283, Dec. 2021, doi: 10.3390/s21248283.
- [27] S. Yeo, Y. Naing, T. Kim, and S. Oh, “Achieving balanced load distribution with reinforcement learning-based switch migration in distributed SDN controllers,” *Electronics (Switzerland)*, vol. 10, no. 2, pp. 1–16, Jan. 2021, doi: 10.3390/electronics10020162.
- [28] M. Waqar and A. Kim, “Performance improvement of ethernet-based fronthaul bridged networks in 5G cloud radio access networks,” *Applied Sciences (Switzerland)*, vol. 9, no. 14, p. 2823, Jul. 2019, doi: 10.3390/app9142823.

## BIOGRAPHIES OF AUTHORS



**Nagaraju Tumakuru Anadanaiah**     obtained bachelor of engineering in electronics and communication. He has completed master of technology in computer network engineering from V.T.U, Belgavi. He is currently working as an Assistant Professor in Govt. Engg. College Ramanagara, Karnataka. He can be contacted at: nagarajuta76@gmail.com.



**Malode Vishwanatha Panduranga Rao**     obtained his Ph.D. degree in computer science from National Institute of Technology Karnataka, Mangalore, India. He has completed a master of technology in computer science and bachelor of engineering in electronics and communication engineering. He is currently working as professor in Jain (Deemed to be University) Bengaluru, India. His research interests are in the field of real-time and embedded systems - internet of things. He has published various research papers in journal and conferences across India, also in the IEEE international conference in Okinawa, Japan (visited) 2008. He has authored two reference books on Linux Internals. He is the Life member of Indian Society for Technical Education and IAENG. Now from past three years, he has published 12 indian patents and three patents are stepping towards grant status. One research scholar under his guidance was awarded a Ph.D. degree. He can be contacted at email: r.panduranga@jainuniversity.ac.in.