International Journal on Science and Technology (IJSAT)

E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Intelligent Path Selection and Load Distribution in Software-Defined Networks: A Critical Review and Conceptual Framework

Prof. Nirzari Patel

Assistant Professor
Information Technology Department
Vidush Somany Institute of Technology and Research, Kadi Sarva Vishwavidyalaya
patelnirzari84@gmail.com

Abstract:
Traffic in modern networks is bursty and heterogeneous, and software-defined networking (SDN) offers a centralized place to react to it. Yet many deployments still forward traffic along fixed shortest paths, so some links saturate while others stay idle. This paper critically reviews research from 2018 to 2022 on path selection, traffic engineering and load distribution in SDN. The reviewed work is grouped into constrained shortest-path routing, learning-based prediction, reinforcement and deep reinforcement learning, controller load balancing, and supporting surveys on fault handling, flow tables and programmable data planes. A comparison table contrasts the studies by technique, focus, strength and limitation. The analysis shows that learning-based routing adapts well but is costly to train and hard to explain, while classical methods are fast but blind to load. Building on this, the paper outlines a conceptual, step-by-step routing and load-distribution procedure that combines flow-size awareness, multi-criteria path scoring and selective rerouting. The procedure is presented as a design proposal for future implementation and is not evaluated here. The paper ends with conclusions and a list of future research directions.

Keywords: Software-defined networking, traffic engineering, path selection, load distribution, reinforcement learning, multi-criteria decision making, review.

I. Introduction

Conventional networks tie forwarding and control together inside every device. Changing a routing policy therefore means touching many boxes, and the network has no single view of its own load. Software-defined networking removes this coupling. A logically central controller computes decisions, and switches simply apply the match-action rules it installs through a southbound protocol such as OpenFlow. Overviews of the field describe this architecture and its main building blocks [16], [18].

The global view is most useful for routing. A controller can see every link, estimate demand, and install a path for each flow. In practice, however, the default behavior in many controllers is still hop-count shortest-path forwarding, which sends traffic along the same few paths regardless of how busy they are. When a few large flows collide on one link, queues build, packets are dropped and applications suffer,

while parallel links carry little traffic. Quality-of-service (QoS) routing was studied long before SDN, and its algorithms have been revisited for the SDN setting [1].

Two trends have shaped recent work. First, the controller itself can host learning components, and surveys describe a move toward data-driven, adaptive networks [6]. Second, learning methods, especially reinforcement learning, have been applied directly to routing and traffic engineering [9], [10], [14], [15]. These methods promise adaptation without an explicit traffic model, but they bring costs: training time, careful reward design and weak interpretability.

This review has three objectives: to organize SDN path-selection and load-distribution research of 2018–2022 by method; to compare the studies critically; and to derive a conceptual procedure that addresses the limitations found. The contributions are:

1. A grouped, critical review of twenty studies, with the practical trade-offs of each group.
2. A comparison table of technique, focus, strength and limitation.
3. A list of research gaps drawn from the comparison.
4. A conceptual, step-by-step routing and load-distribution procedure, with its complexity discussed in general terms.

The rest of the paper is organized as follows. Section II reviews the literature. Section III gives the comparison table. Section IV states the research gaps. Section V presents the conceptual procedure. Section VI concludes and Section VII outlines future work.

II. Literature Work

Scope. The review covers papers from January 2018 to December 2022 chosen by the author for relevance and venue quality. It is not a systematic search. Metaheuristic methods such as genetic algorithms, particle swarm, ant colony and artificial bee colony optimization are widely used for network optimization, but this review does not include a dedicated 2018–2022 SDN study of each; they are discussed only in general terms and flagged as a coverage limit.

A. Classical and constrained shortest-path routing

Dijkstra's algorithm and Bellman-Ford remain the building blocks of path computation. Guck et al. compare unicast QoS routing algorithms for SDN and show that constrained shortest-path methods differ greatly in how well they respect delay and bandwidth limits and in how long they take to run [1]. The lesson is practical: a heuristic that finds good constrained paths quickly is often preferable to an exact method that scales poorly. A limit of this line of work is that the algorithms are evaluated on given link states and are not coupled to online measurement.

B. Metaheuristic optimization

Genetic, particle swarm, ant colony and bee colony algorithms search large solution spaces and handle several objectives at once. Their weakness for SDN control is timing. They iterate over populations, their convergence time varies from run to run, and their parameters need tuning. For a controller that must place a path within a short window, this makes them better suited to periodic offline re-optimization than to per-flow decisions. This general assessment is stated without a specific citation, in line with the coverage limit above.

C. Learning-based traffic prediction and classification

Machine learning is used to classify traffic and predict its volume so that the controller can act early. Mohammed et al. apply machine-learning and deep-learning models to traffic classification and prediction in SDN [8]. Such work improves the information available to routing but does not itself choose paths, and

its benefit depends on the quality and representativeness of the training data. Broader reviews of data-driven softwarized networks highlight the same dependency on data and the difficulty of keeping models current [6].

D. Reinforcement and deep reinforcement learning for routing and traffic engineering

This is the most active area. Chen et al. propose RL-Routing, a deep reinforcement learning routing scheme for SDN that uses measured network state to choose paths [9]. Liu et al. address online routing for requests with different service requirements using deep reinforcement learning [14]. Troia et al. apply deep reinforcement learning to traffic engineering in SD-WAN [15]. Zhang et al. take a more selective view: CFR-RL learns which critical flows to reroute so that maximum link utilization drops while most flows stay untouched [10]. Yao et al. describe an architecture for self-learning control in SDN built around deep reinforcement learning [3]. Mao et al. use deep learning to compute routes in real time for software-defined communication systems [19].

Compared with classical routing, these schemes adapt to changing load. Their common weaknesses are the training effort, sensitivity to the reward function, uncertain behavior on topologies or traffic patterns unseen during training, and limited explanation of why a path was chosen. CFR-RL is notable because rerouting only a few flows limits disruption and shrinks the action space, an idea that carries over to simpler non-learning designs.

E. Load balancing, controller placement and flow tables

Load is a problem on the control plane as well as the data plane. Sun et al. propose MARVEL, which uses multi-agent reinforcement learning to balance load across controllers [11]. Surveys of multi-controller SDN discuss placement, consistency and scalability trade-offs [2]. Flow-table capacity is another resource: Mu et al. use reinforcement learning to manage flow entries [4], and Isyaku et al. survey flow-table management with its performance and security issues [12]. These studies show that resource allocation in SDN covers bandwidth, controller capacity and switch memory, and that a routing scheme which ignores table space can create new bottlenecks.

F. Application and supporting surveys

Rego et al. use SDN-based traffic management to prioritize emergency traffic in smart cities [5], illustrating policy-driven routing. Yu et al. survey fault management in SDN [7], and Ali et al. survey link-failure recovery [13]; both matter because a routing scheme must remain correct when links fail. Ibrar et al. use machine learning to limit access-control policy violations after link failure in hybrid SDN [20]. Priyadarsini and Bera survey SDN architecture, traffic management, security and placement [16]; Hussain et al. categorize SDN work and future directions [18]; and Kaur et al. review programmable data planes based on P4 [17], which widen what switches can measure and do.

III. Comparison Table

Table I. Comparison of reviewed studies (strength and limitation are the reviewer's assessment of the scope of each paper; confirm against the original)

Ref.	Year	Technique	Main focus	Strength	Limitation
[1]	2018	Constrained shortest-path algorithms	QoS routing comparison	Clear comparison of speed and constraint handling	Static evaluation, no online learning

[2]	2018	Survey	Multi-controller SDN	Covers placement and consistency	No routing algorithm proposed
[3]	2018	Deep RL architecture	Self-learning control	Integrates sensing and learning	Architecture-level emphasis
[4]	2018	Reinforcement learning	Flow-entry management	Targets table-space limits	Not path selection
[5]	2018	SDN traffic management	Emergency traffic in smart cities	Policy-driven prioritization	Application-specific
[6]	2019	Review	Data-driven softwarized networks	Broad view of opportunities	General, not algorithm-specific
[7]	2019	Survey	Fault management	Detailed failure handling	No routing algorithm proposed
[8]	2019	ML and deep learning	Traffic classification and prediction	Anticipates demand	Prediction only, data dependent
[9]	2020	Deep RL	QoS-aware SDN routing	Adapts to measured state	Training cost, limited explanation
[10]	2020	RL for critical flows	Traffic engineering	Reroutes few flows, low disruption	Needs training, depends on flow selection quality
[11]	2020	Multi-agent RL	Controller load balancing	Distributed decisions	Control plane only
[12]	2020	Survey	Flow-table management	Performance and security view	No routing algorithm proposed
[13]	2020	Survey	Link-failure recovery	Covers recovery approaches	Not load-aware routing
[14]	2021	Deep RL	Online routing, mixed service types	Handles varied requirements	Training and generalization concerns
[15]	2021	Deep RL	Traffic engineering in SD-WAN	Centralized adaptive control	Training cost, WAN-specific
[16]	2021	Survey	Architecture, traffic, security	Wide coverage	Limited algorithm detail
[17]	2021	Survey	P4 data planes	Shows richer data-plane control	Not routing-specific
[18]	2022	Survey	Categories and directions	Up-to-date taxonomy	Limited algorithm detail
[19]	2021	Deep learning	Real-time route computation	Fast inference after training	Needs training data and retraining

[20] 2020	Machine learning	Policy compliance after link failure	Targets a concrete failure problem	Hybrid-SDN specific

IV. Research Gaps

1. **Speed versus adaptivity.** Classical algorithms are fast but load-blind [1]; learning methods adapt but need training [9], [14], [15]. Few designs offer adaptation with predictable, low computation.
2. **Flow-size awareness.** Only some studies treat large and small flows differently [10], although flow size strongly affects congestion.
3. **Selective action.** Rerouting everything on every update causes churn; limiting changes to a few flows is promising but rarely combined with simple scoring rules [10].
4. **Joint resource view.** Bandwidth, controller load and flow-table space are usually studied separately [4], [11], [12].
5. **Robustness and failure.** Routing papers seldom examine behavior under link failure alongside load [7], [13].
6. **Evaluation practice.** Surveys point to limited benchmarks and reproducibility in the field [6], [18].

V. Proposed Algorithm Steps (Conceptual)

The procedure below, called here CALM (congestion-aware, load-managed routing), is a design proposal derived from the gaps above. It has not been implemented or tested, and no performance claims are made for it. It targets gaps 1–3 and 5 with a lightweight, explainable design, and leaves a learning component as an optional extension.

Inputs: network graph G = (V, E); link capacities; monitoring period T; criteria weights; flow-size threshold; congestion threshold with persistence count; hysteresis margin.

Procedure:

1. **Discover topology.** Build G from controller topology discovery and update it on link up/down events.
2. **Collect state.** Every T seconds read port and flow counters to obtain, for each link, utilization, loss and delay estimates.
3. **Smooth the measurements.** Apply exponential smoothing so that short spikes do not trigger action; optionally replace it with a short-horizon predictor [8].
4. **Classify flows.** Label a flow as large if its byte count exceeds the threshold, otherwise small.
5. **Generate candidates.** For each new flow, compute k loop-free low-cost paths with a shortest-path method [1] and discard paths that violate failed or reserved links.
6. **Score paths with multi-criteria decision making.** Rate each candidate on delay, loss, residual bandwidth, maximum link utilization and hop count. Normalize the criteria, weight them, and rank paths by closeness to the ideal solution: $C(p) = D^{-}(p) / (D^{+}(p) + D^{-}(p))$, where D^{+} and D^{-} are distances to the best and worst values.
7. **Assign by flow size.** Place a large flow on the highest-ranked path with enough headroom; spread small flows across the top-ranked paths.
8. **Detect congestion.** Mark a link congested when its smoothed utilization exceeds the threshold for several consecutive periods.

9. **Reroute selectively.** From the flows crossing a congested link, move only the largest few to better-ranked paths, and only if the score gain exceeds the hysteresis margin, in the spirit of critical-flow rerouting [10].
10. **Handle failure.** On a link failure, recompute candidates for affected flows only and reinstall their rules [7], [13].
11. **Adapt weights (optional).** Adjust the criteria weights slowly according to observed congestion events, or delegate this to a learning agent [9], [15].
12. **Output.** Install flow rules along the chosen paths with suitable timeouts and record decisions for audit.

Complexity. The cost per new flow is dominated by candidate generation, which is polynomial in the number of nodes and links and grows linearly with k; scoring adds a term linear in k times path length. Periodic monitoring costs time proportional to the number of links. This is bounded and predictable, unlike population-based search.
Expected trade-offs (to be tested, not claimed): explainable decisions and low computation, at the price of hand-set weights and thresholds that need tuning.

VI. Conclusion

This review examined SDN path-selection and load-distribution research from 2018 to 2022. Classical constrained routing is fast but ignores live load, learning-based routing adapts but costs training effort and clarity, and controller and flow-table studies show that resources beyond bandwidth must be managed. The comparison table and gap analysis motivated a conceptual procedure that scores paths on several criteria, treats large and small flows differently, and reroutes only a few flows when congestion persists. The procedure remains a proposal until it is implemented and evaluated against established baselines.

VII. Future Work

1. Implement and evaluate the conceptual procedure in an emulator against shortest-path and load-balancing baselines, with repeated runs.
2. Add a learning component for weight adaptation, such as deep reinforcement learning [14], [15].
3. Extend to multi-controller deployments [2], [11].
4. Study SDN integration with IoT and edge networks.
5. Add security-aware routing that avoids suspicious paths.
6. Include energy-efficient routing, for example by consolidating traffic.
7. Use dynamic traffic prediction to reroute before congestion [8].
8. Cover metaheuristic methods with dedicated SDN studies, closing the coverage gap noted in Section II.
9. Validate on hardware testbeds, including P4 switches [17].

REFERENCES:

[1] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer, "Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 388–415, 2018.

[2] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *J. Netw. Comput. Appl.*, vol. 103, pp. 101–118, 2018.
[3] H. Yao, T. Mai, X. Xu, P. Zhang, M. Li, and Y. Liu, "NetworkAI: An intelligent network architecture for self-learning control strategies in software defined networks," *IEEE Internet Things J.*, vol. 5, no. 6, 2018.
[4] T.-Y. Mu, A. Al-Fuqaha, K. Shuaib, F. M. Sallabi, and J. Qadir, "SDN flow entry management using reinforcement learning," *ACM Trans. Auton. Adapt. Syst.*, vol. 13, no. 2, 2018.
[5] A. Rego, L. Garcia, S. Sendra, and J. Lloret, "Software defined network-based control system for an efficient traffic management for emergency situations in smart cities," *Future Gener. Comput. Syst.*, vol. 88, 2018.
[6] W. Kellerer et al., "Adaptable and data-driven softwarized networks: Review, opportunities, and challenges," *Proc. IEEE*, vol. 107, no. 4, pp. 738–755, 2019.
[7] Y. Yu et al., "Fault management in software-defined networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 349–392, 2019.
[8] A. R. Mohammed, S. A. Mohammed, and S. Shirmohammadi, "Machine learning and deep learning based traffic classification and prediction in software defined networking," in *Proc. IEEE Int. Symp. Meas. Netw. (M&N)*, 2019.
[9] Y.-R. Chen, A. Rezapour, W.-G. Tzeng, and S.-C. Tsai, "RL-Routing: An SDN routing algorithm based on deep reinforcement learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 3185–3199, 2020.
[10] J. Zhang, M. Ye, Z. Guo, C.-Y. Yen, and H. J. Chao, "CFR-RL: Traffic engineering with reinforcement learning in SDN," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2249–2259, 2020.
[11] P. Sun, Z. Guo, G. Wang, J. Lan, and Y. Hu, "MARVEL: Enabling controller load balancing in software-defined networks with multi-agent reinforcement learning," *Comput. Netw.*, vol. 177, art. 107230, 2020.
[12] Y. Isyaku, M. Zulkifli, and K. Kamin, "Software defined networking flow table management of OpenFlow switches performance and security challenges: A survey," *Future Internet*, vol. 12, no. 9, art. 147, 2020.
[13] J. Ali, G.-M. Lee, B.-H. Roh, D. K. Ryu, and G. Park, "Software-defined networking approaches for link failure recovery: A survey," *Sustainability*, vol. 12, no. 10, art. 4255, 2020.
[14] W.-X. Liu et al., "DRL-OR: Deep reinforcement learning-based online routing for multi-type service requirements," in *Proc. IEEE INFOCOM*, 2021.
[15] S. Troia, F. Sapienza, L. Varé, and G. Maier, "On deep reinforcement learning for traffic engineering in SD-WAN," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2198–2212, 2021.
[16] M. Priyadarsini and P. Bera, "Software defined networking architecture, traffic management, security, and placement: A survey," *Comput. Netw.*, vol. 192, art. 108047, 2021.
[17] S. Kaur, K. Kumar, and N. Aggarwal, "A review on P4-programmable data planes: Architecture, research efforts, and future directions," *Comput. Commun.*, 2021.
[18] M. Hussain, N. Shah, R. Amin, S. S. Alshamrani, A. Alotaibi, and S. M. Raza, "Software-defined networking: Categories, analysis, and future directions," *Sensors*, vol. 22, no. 15, art. 5551, 2022.

[19] B. Mao, F. Tang, Z. M. Fadlullah, and N. Kato, "An intelligent route computation approach based on real-time deep learning strategy for software defined communication systems," *IEEE Trans. Emerg. Topics Comput.*, 2021 — **verify details**.

[20] M. Ibrar et al., "PrePass-Flow: A machine learning based technique to minimize ACL policy violation due to links failure in hybrid SDN," *Comput. Netw.*, 2020 — **verify details**.