Intelligent Deflection Routing in Buffer-Less Networks

Soroush Haeri, *Student Member, IEEE* and Ljiljana Trajković, *Fellow, IEEE*

***Abstract*—Deflection routing is employed to ameliorate packet loss caused by contention in buffer-less architectures such as optical burst-switched (OBS) networks. The main goal of deflection routing is to successfully deflect a packet based only on a limited knowledge that network nodes possess about their environment.**

In this paper, we present a framework that introduces intelligence to deflection routing (iDef). iDef decouples the design of the signaling infrastructure from the underlying learning algorithm. It consists of a signaling and a decision-making module. Signaling module implements a feedback management protocol while the decision-making module implements a reinforcement learning algorithm. We also propose several learning-based deflection routing protocols, implement them in iDef using the ns-3 network simulator, and compare their performance.

***Index Terms*—Associative learning, buffer-less networks, decision-making, deflection routing, reinforcement learning.**

I. Introduction

DEFLECTION routing is a viable contention resolution scheme that may be employed in buffer-less networks such as optical burst-switched (OBS) networks [1]. Contention occurs when according to a routing table, multiple arriving traffic flows at a node need to be routed through a single outgoing link. In this case, only one flow is routed through the optimal link defined by the routing table. In the absence of a contention resolution scheme, the remaining flows are discarded because the node possesses no buffers. Instead of buffering or discarding packets, deflection routing helps to temporarily deflect them away from the path that is prescribed by the routing table.

Deflection routing may benefit from the random nature of reinforcement learning algorithms. A deflection routing algorithm coexists in the network along with an underlying routing protocol that usually generates a significant number of control signals. Therefore, it is desired that deflection routing protocols generate few control signals. Reinforcement learning algorithms enable a deflection routing protocol to generate viable deflection decisions by adding a degree of randomness to the decision-making process.

Manuscript received March 23, 2014; revised May 13, 2014, June 30, 2014, and September 19, 2014; accepted September 20, 2014. Date of publication December 18, 2014; date of current version January 13, 2015. This work was supported by the Natural Sciences and Engineering Research Council of Canada under Grant 216844-13. This paper was recommended by Associate Editor T. Vasilakos.

The authors are with the School of Engineering Science, Simon Fraser University, Vancouver, BC V5A 1S6, Canada (e-mail: shaeri@sfu.ca; ljilja@sfu.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2014.2360680

Reinforcement learning-based algorithms were proposed in the early days of the Internet to develop routing policies [2]–[4]. Q-learning [5] is a reinforcement learning algorithm that has been employed for generating routing policies. The Q-routing algorithm [2] requires that nodes locally make their routing decisions. Each node learns a local deterministic routing policy using the Q-learning algorithm. Generating the routing policies locally is computationally less intensive. However, the Q-routing algorithm does not generate an optimal routing policy in networks with low loads nor does it learn new optimal policies in cases when network load decreases. The Predictive Q-routing algorithm [3] addresses these shortcomings by recording the best experiences learned, which may then be reused to predict traffic behavior. Packet routing algorithms in large networks such as the Internet have to consider the business relationships between Internet Service Providers. Therefore, in such environment, randomness is not a desired property of a routing algorithm. Consequently, Internet routing algorithms have not employed reinforcement learning because of its inherent randomness.

The contributions of this paper are:

1) We develop a framework named iDef that simplifies implementation and testing of deflection routing protocols by employing a modular architecture. The ns-3 [6] implementation of iDef is made publicly available [7].
2) We introduce the novel Node Degree Dependent (NDD) signaling algorithm [8]. The complexity of the algorithm only depends on the degree of the node that is NDD compliant while the complexity of the other currently available reinforcement learning-based deflection routing algorithms depends on the size of the network. Therefore, NDD is better suited for larger networks. Simulation results show that NDD-based deflection routing algorithms scale well with the size of the network and perform better than the existing algorithms. Furthermore, the NDD signaling algorithm generates fewer control signals compared to the existing algorithms.
3) For the decision-making, we propose a feed-forward neural network (NN) and a feed-forward NN with episodic updates (ENN). They employ a single hidden layer NN that updates its weights using an associative learning algorithm. Currently available reinforcement learning-based deflection routing algorithms employ Q-learning, which does not utilize efficiently the gathered feedback signals. NN and ENN decision-making algorithms address the deficiency of Q-learning by introducing a single hidden layer NN to generate deflection

decisions. The NN-based deflection routing algorithms achieve better results than Q-learning-based algorithms in networks with low to moderate loads. Efficiently utilizing control signals in such cases is important because the number of deflections is small and a deflection routing algorithm receives fewer feedback signals.

4) The proposed NDD signaling algorithm with NN, ENN, and Q-learning-based decision-making modules was incorporated into three reinforcement learning-based deflection routing protocols named NN-NDD, ENN-NDD, and Q-NDD. We implement these protocols within the iDef framework and compare their performance with the existing Reinforcement Learning Deflection Routing Scheme (RLDRS) [9] and Predictive Q-learning Deflection Routing (PQDR) [10].

The remainder of this manuscript is organized as follows. In Section II, we describe buffer-less network architectures and contention in such networks. Reinforcement learning is presented in Section III. In Section IV, we introduce deflection routing as a contention resolution scheme and provide a brief survey of work related to applications of reinforcement learning in deflection routing. We present the iDef framework in Section V and the NDD signaling algorithm in Section VI. Designs of the NN and ENN decision-making modules are presented in Section VII. Their performance is evaluated in Section VIII. We conclude with Section IX.

II. Buffer-Less Architecture and Contention

Nodes in buffer-less networks do not posses memory (buffer) to store packets. Buffers are usually implemented as first-in-first-out (FIFO) queues that are used to store packets contending to be forwarded to the same outgoing link. Examples of buffer-less architectures are OBS networks and network-on-chips. OBS is a technology designed to share optical fiber resources across data networks [1]. Other optical switching technologies for data communication such as Synchronous Optical Network (SONET) and Synchronous Digital Hierarchy (SDH) [11] reserve the entire light-path from a source to a destination. Even though a light-path is not fully utilized, it may not be shared unless its reservation is explicitly released. The OBS technology overcomes these limitations. Switching in OBS networks is performed optically, allowing the optical/electrical/optical conversions to be eliminated in the data plane. This permits high capacity switching with simpler switching architecture and lower power consumption [12].

In OBS networks, data are optically switched. At an ingress node of such a network, multiple packets are aggregated into one optical burst. Before transmitting the burst, a burst header packet (BHP) is created and sent ahead of the burst with an offset time t_{offset}. The BHP contains information needed to perform OBS switching and IP routing, such as the burst length t_{length}, offset time t_{offset}, and the destination IP address. When an OBS node receives a BHP, it has t_{offset} time to locate the outgoing link l_o in the routing table, reserve the link for the burst to pass through, and reconfigure the optical cross-connect (OXC) module that connects the incoming and the outgoing links.

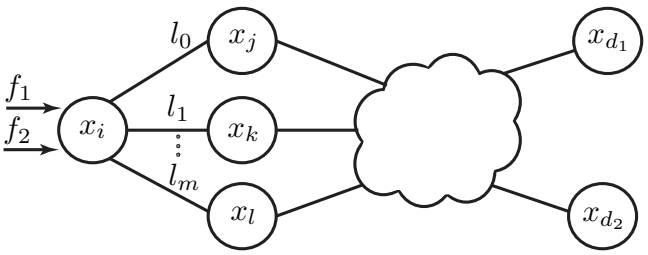

Fig. 1. A network with n buffer-less nodes.

Consider an arbitrary buffer-less network shown in Fig. 1, where $\mathcal{N} = \{x_1, x_2, \ldots, x_n\}$ denotes the set of all network nodes. Node x_i routes the incoming traffic flows f_1 and f_2 to the destination nodes x_{d_1} and x_{d_2}, respectively. According to the shortest path routing table stored in x_i, both flows f_1 and f_2 should be forwarded to node x_j via the outgoing link l_0. In this case, the flows f_1 and f_2 are contending for the outgoing link l_0. The node x_i forwards flow f_1 through l_0 to the destination x_{d1}. However, in the absence of a contention resolution scheme, the flow f_2 is discarded because node x_i is unable to buffer it. Deflection routing [13] is a contention resolution scheme that may be employed to reduce packet loss. Contention between the flows f_1 and f_2 is resolved by routing f_1 through the preferred outgoing link l_0 while routing f_2 through alternate outgoing link $l \in \mathcal{L}\backslash\{l_0\}$, where the set $\mathcal{L}$ denotes the set of all outgoing links connected to x_1. Various other methods have been proposed to resolve contention. Wavelength conversion [14], fiber delay lines [15], and control packet buffering [16] are among the contention resolution schemes that are applicable to optical networks.

III. Reinforcement Learning

Reinforcement learning algorithms learn situation-to-action mappings with the main objective to maximize numerical rewards. These algorithms may be employed by agents that learn to interact with a dynamic environment through trial-and-error [17]. Reinforcement learning consists of three abstract events irrespective of the learning algorithm: an agent observes the state of the environment and selects an appropriate action; the environment generates a reinforcement signal and transmits it to the agent; the agent employs the reinforcement signal to improve its subsequent decisions. Therefore, a reinforcement learning agent requires information about the state of the environment, reinforcement signals from the environment, and a learning algorithm.

Q-learning [5] is a reinforcement learning algorithm that has been employed for path selection in deflection routing. The algorithm maintains a Q-value $Q(s, a)$ in a Q-table for every state-action pair. Let s_t and a_t denote the encountered state and the action executed by an agent at a time instant t, respectively. Furthermore, let r_{t+1} denote the reinforcement signal that the environment has generated for performing action a_t in state s_t. When the agent receives the reward r_{t+1}, it updates the

Q-value that corresponds to the state s_t and action a_t as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \times \Big[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \Big], \quad (1)$$

where $0 < \alpha \leq 1$ is the learning rate and $0 \leq \gamma < 1$ is the discount factor.

Enhancing a node in buffer-less networks with a reinforcement learning agent that generates deflection decisions requires three components: function that maps a collection of the environment variables to an integer (state); decision-making algorithm that selects an action based on the state; signaling mechanism for sending, receiving, and interpreting the feedback signals. The decision-making instances in deflection routing are intermittent. Hence, a learning algorithm need not consider the effect of all possible future system trajectories for making its decisions. Agents that do not consider future system trajectories for decision-making are known as immediate reward or associative learning agents.

IV. Deflection Routing by Reinforcement Learning

Deflection routing has attracted significant attention [18]–[20] as a viable method to resolve contention in buffer-less networks. Studies show that multiple deflections may significantly improve utilization in networks with multiple alternate routes such as fully meshed networks [21]. Performance evaluation of various deflection routing algorithms using scale-free Barabási-Albert [22] topologies, which resemble the autonomous system-level view of the Internet, shows that deflection routing effectively reduces the burst loss probability in optical networks [23].

Deflection routing algorithms generate deflection decisions based on a deflection set $\mathcal{D}$, which includes all viable alternate links available for deflection. The size of the set $\mathcal{D}$ determines the flexibility of deflection module in resolving contention. One approach includes all available links in the set $\mathcal{D}$, which would maximize the flexibility while possibly introducing routing loops. Another approach includes only outgoing links that lie on the shortest path, which avoids routing loops while reducing the flexibility in decision-making [24]. Several algorithms have been proposed to populate large deflection sets $\mathcal{D}$ while ensuring no routing loops [25], [26]. The proposed NDD signaling algorithm begins by including all outgoing links in the deflection set $\mathcal{D}$. As time progresses and deflection module receives feedback signals, the probability of selecting alternate links that may result in routing loops decreases.

Performance analysis of deflection routing based on random decisions shows that random deflection may effectively reduce blocking probability and jitter in networks with light traffic loads [27]. Advanced deflection routing algorithms improve the quality of decision-making by enabling neighboring nodes to exchange traffic information [9], [28]–[31]. This information provides to the deflection module an integral neighborhood view for decision-making. Heuristic methods may then be employed to make deflection decisions based on the collected traffic information. Reinforcement learning, which has been implemented in the proposed NN-NDD and ENN-NDD algorithms, provides a systematic framework for processing the gathered information. Various other deflection routing protocols based on reinforcement learning [8]–[10], [30] employ the Q-learning algorithm or its variants.

The Q-learning Path Selection algorithm [30] calculates a priori a set of candidate paths $P = \{p_1, \ldots, p_m\}$ for tuples (s_i, s_j), where $s_i, s_j \in S$ and $S = \{s_1, \ldots, s_n\}$ denotes the set of all edge nodes in the network. The Q-table stored in the i^{th} edge node maintains a Q-value for every tuple (s_j, p_k), where $s_j \in S \backslash \{s_i\}$ and $p_k \in P$. The sets S and P are states and actions, respectively. The Q-value is updated after each decision is made and the score of the path is reduced or increased depending on the received rewards. The algorithm does not specify a signaling method or a procedure for handling the feedback signals. RLDRS [9] also employs the Q-learning algorithm for deflection routing. The advantages of RLDRS are its precise signaling and rewarding procedures.

Decisions that repeatedly receive poor rewards have low Q-values. Certain poor rewards might be due to a transient link failure or congestion and, hence, recovery and reselection of such decisions may improve the decision-making. Q-learning-based deflection routing algorithms do not provide a procedure to reselect the paths that have low Q-values as a consequence of transient network conditions. The PQDR algorithm [10] enables a node to recover and reselect such paths and improve its decision-making ability. The PQDR algorithm combines the Predictive Q-routing algorithm [3] and RLDRS to optimally deflect contending flows. When deflecting a traffic flow, the PQDR algorithm stores in a Q-table the accumulated reward for each deflection decision. It also recovers and reselects decisions that are not well rewarded and have not been used over a period of time.

The Q-learning Path Selection algorithm, RLDRS, and PQDR have two drawbacks: they are not scalable and their underlying learning algorithms are inefficient.

1) Scalability: Q-learning Path Selection algorithm and RLDRS are not scalable because their space complexity depends on the network size. For example, the size of the Q-table generated by the Q-learning Path Selection algorithm [30] depends on the size of the network and the set of candidate paths. Therefore, it may be infeasible to store a large Q-table emanating from a large network.

2) Learning Deficiency: Q-learning is the only learning algorithm that has been employed for the deflection routing [8]–[10], [30]. The learning algorithm used in PQDR [10] is a variant of Q-learning and, hence, it inherits some of its deficiencies. Even though Q-learning guarantees eventual convergence to an optimal policy when finding the best action set, it does not efficiently use the gathered data. Therefore, it requires gaining additional experience to achieve good results [17].

The proposed NDD signaling algorithm addresses the scalability issue of the current reinforcement learning-based deflection routing algorithms. Its complexity depends only on a node degree. The NN and ENN decision-making modules are introduced to address the learning deficiency of Q-learning. They learn based on the REINFORCE associative learning algorithm [32], [33] that utilizes the gained experience more

efficiently than Q-learning.

V. The iDef Framework

The proposed iDef framework is designed to facilitate development of reinforcement learning-based deflection routing protocols. We implemented the iDef framework in the ns-3 network simulator. In iDef, a reinforcement learning-based deflection routing algorithm is abstracted using mapping, decision-making, and signaling modules. iDef is designed to minimize the dependency among its modules. This minimal dependency enables implementation of portable deflection routing protocols within the iDef framework, which enables modules to be replaced without changing the entire design. For example, replacing the decision-making module requires no changes in the implemented signaling module.

A deflection manager glues together the iDef modules. It has access to the OBS network interface cards and the IP routing table. The deflection manager makes iDef portable by eliminating communication among mapping, decision-making, and signaling modules. The burst header messages received by a node are passed to the deflection manager. The deflection manager inspects the IP routing table for the next hop and then checks the status of the optical interfaces. If the desired optical interface is available, the optical cross-connects are configured according to the path defined by the IP routing table. If the interface is busy, the deflection manager passes the environment variables such as destination of the burst, output links blocking state, and the next hop on the shortest path, to the mapping module.

The *mapping module* maps all or a subset of the received variables to an integer called the *state*. For example, in the proposed NDD signaling algorithm, one possibility of such mapping is to append the binary ID of the port number obtained from the routing table to a string of 0s and 1s that represents the outgoing links status. This binary string may then be converted to an integer.

The *decision-making module* implements the learning algorithm. Therefore, the statistics, the history, and other required information for decision-making are stored in this module. It implements two main functions that are used by the deflection manager: a function that generates actions given a state and a function that updates the statistics when a reinforcement signal is received. The mapped state is passed to the decision-making module where an alternate output link (action) is selected for the burst. The generated decisions are then passed to the deflection manager. The signaling module passes the received reinforcement signals to the decision-making module where they are used for statistic updates.

The *signaling module* adds header fields to the deflected bursts. It also inspects the burst headers received by the deflection manager for special deflection header fields and tags. It assembles and sends feedback messages when required. Upon receiving a feedback message, the signaling module interprets the reinforcement signal and translates it to a scalar reward. This reward is then used by the deflection manager to enhance the decision-making module.

VI. The Node Degree Dependent Signaling Algorithm

We describe here the proposed NDD [8] signaling algorithm and the messages that need to be sent in order to enhance a buffer-less node with a decision-making ability. The NDD algorithm provides a signaling infrastructure that nodes may require in order to learn and then optimally deflect packets in a buffer-less network.

We consider a buffer-less network with N nodes. All nodes are iDef compatible and, hence, possess mapping, decision-making, and signaling modules. The headers of the packets received by a node are passed to the signaling module. The module inspects the routing table for the next hop and then checks the status of the network interfaces. If the desired interface is available, the packet is routed according to the path defined by the routing table. If the interface is busy and the packet has not been deflected earlier by any other node, the current states of the network interfaces and the output port obtained from the routing table are passed to the mapping module. The mapping module maps these inputs to a unique representation known as the system *state*. It then passes this information (*state*) to the decision-making module. The *state* representation depends on the underlying learning algorithm. Therefore, it may change based on the design of the decision-making module. Various decision-making modules require specifically designed compatible mapping modules. For example, Q-learning-based decision-making module requires a mapping module that transforms the current states of the network interfaces and the output port suggested by the routing table to a real number while NN and ENN-based decision-making modules require binary vectors. The mapping module maps the states of the network interfaces to an ordered string of 0s and 1s, where idle and busy interfaces are denoted by 0 and 1, respectively.

The decision-making module returns the index of the best network interface for deflecting the packet based on the current *state*. The signaling module adds the following information to the packet header: a unique *ID* number used to identify the feedback message that pertains to a deflection; the address of the node that initiated the deflection, to be used by other nodes as the destination for the feedback messages; a deflection hop counter *DHC*, which is incremented each time other nodes deflect the packet.

When a packet is to be deflected at a node for the first time, the node records the current time as the deflection time *DfT* along with the *ID* assigned to the packet. The Drop Notification *DN* timer is initiated and the packet is deflected to the network interface that is selected by the decision-making module. The maximum value of the DN timer is set to DN_{max}, which indicates expiration of the timer. The purpose of the timer is to reduce the number of feedback signals.

After a deflection decision is made, the decision-making module waits for the feedback. It makes no new decisions during this idle interval. The deflected packet is discarded when either it reaches a fully congested node or its *DHC* reaches the maximum permissible number of deflections DHC_{max}.

The node that discards the deflected packet assembles a

feedback message consisting of the packet *ID*, *DHC*, and the time instant *DrT* when the packet was discarded. The feedback message is then sent to the node that initiated the deflection.

When the node that initiated the deflection receives the feedback message, it calculates the total travel time *TTT* that the packet has spent in the network after the first deflection:

$$TTT = DrT - DfT. \tag{2}$$

The *TTT* and *DHC* values are used by the decision-making module to update its statistics. If no feedback message is received until the *DN* timer expires, the node assumes that the packet has arrived successfully to its destination. The node may then update its decision-making module with the reinforcement signal having $TTT = 0$ and $DHC = 0$. A decreasing function with the global maximum at $(0, 0)$ is used as a reward function to map *TTT* and *DHC* to a real value r.

A node records the best action selected by the decision-making module. These records are used if a node needs to deflect a packet that has been deflected earlier or during an idle interval.

In order to reduce the excess traffic generated by the number of feedback messages, a node receives feedback messages only when it deflects packets that have not been deflected earlier. Hence, only deflecting such packets enhances the node's decision-making ability.

VII. NNs for Deflection Routing

In this Section, we first describe the REINFORCE algorithm [33]. We then propose the feed-forward NN to be used for generating deflection decisions. The NN decision-making module implements a network of interconnected learning agents. In this Section, the term network refers to an NN.

A. Feed-Forward NNs for Reinforcement Learning

In a network of learning agents with weighted interconnections, the following steps are executed for each decision made:

- network receives an input from its environment;
- the input propagates through the network and it is mapped to an output;
- the output is fed to the environment for evaluation;
- environment evaluates the output signal and generates a reinforcement signal r;
- each network unit modifies its weights based on a learning algorithm using the received reinforcement signal.

Let w_{ij} denote the weight of the connection between agents i and j. The behavior of the agent i is defined by a vector $\mathbf{w}_i$ that contains the weights of all connections to the ith agent. The agent i receives an input vector $\mathbf{x}_i$ from the environment and/or other agents and then maps the input to an output y_i according to a probability distribution function $g_i(\zeta, \mathbf{w}_i, \mathbf{x}_i) = \Pr\{y_i = \zeta \mid \mathbf{w}_i, \mathbf{x}_i\}$ [33]. In case of binary agents where the input is mapped to either 0 or 1 with probabilities p_i and $1-p_i$, respectively, g_i is defined as a Bernoulli semilinear unit:

$$g_i(\zeta, \mathbf{w}_i, \mathbf{x}_i) = \begin{cases} 1 - p_i & if \quad \zeta = 0 \\ p_i & if \quad \zeta = 1 \end{cases}, \tag{3}$$

where

$$\begin{aligned} p_i =& f_i(\mathbf{w}_i^T \mathbf{x}_i) \\ =& f_i\Big(\sum_j w_{ij} x_j\Big). \end{aligned} \tag{4}$$

Function f_i is a differentiable squashing function. A commonly used function is the logistic map:

$$f_i\big(\sum_j w_{ij} x_j\big) = \frac{1}{1 + e^{-\sum_j w_{ij} x_{ij}}}, \tag{5}$$

where x_{ij} is the jth element of the input vector $\mathbf{x}_i$. Agents that use Bernoulli semilinear unit along with the logistic function are called *Bernoulli-logistic* units.

Upon receiving a feedback signal r, the NN updates its weights. The update rule for Bernoulli-logistic units suggested by the REINFORCE algorithm updates the weights w_{ij} as:

$$\Delta w_{ij} = \alpha r (y_i - p_i) x_{ij}, \tag{6}$$

where α is a non-negative rate factor, y_i is the output of the ith agent, p_i is the probability of $y_i = 1$ given the input vector $\mathbf{x}_i$ and weight vector $\mathbf{w}_i$, and x_{ij} is the jth element of the input vector $\mathbf{x}_i$. Let matrix $\mathbf{W}$ denote the collection of weights that determines the behavior of an arbitrary feed-forward network. Under the assumption that the environment's inputs to the network and the received reinforcement signals for any input/output pair are determined by stationary distributions, (6) maximizes the expected value of the reinforcement signal, given the weight matrix $\mathcal{E}\{r \mid \mathbf{W}\}$.

The REINFORCE algorithm utilizes an episodic update rule in order to operate in environments where delivery of the reinforcement signals has unknown delays. A k-episode is defined when a network selects k actions between two consecutive reinforcement receptions. The k-episode is terminated upon receiving a reinforcement signal. The REINFORCE algorithm suggests that a network may update its weighs at the end of the k-episode as:

$$\Delta w_{ij} = \alpha r \sum_{l=1}^{k} [(y_i(l) - p_i(l)) x_j(l-1)]. \tag{7}$$

B. Feed-Forward NN for Deflection Routing with Single Episode Updates

We propose a single-hidden-layer feed-forward network for deflection routing because fewer number of algebraic operations are required when the network selects actions and updates its weights. Furthermore, such networks have reasonable learning capabilities [34], [35]. The feed-forward NN for generating deflection decisions is composed of the input, middle, and output layers, as shown in Fig. 2.

Consider a buffer-less node with n outgoing links. The input layer of such a node consists of two partitions denoted by binary vectors $\mathbf{I}^l = [i_1^l \ \ldots \ i_n^l]$ and $\mathbf{I}^d = [i_d^l \ \ldots \ i_n^d]$. If the k^{th} outgoing link of the node is blocked, i_k^l is set to 1. It is 0 otherwise. If the burst that is to be deflected contends for the j^{th} outgoing link of the node, the j^{th} entry of the input vector $\mathbf{I}^d$ is set to 1 while the remaining elements are set to 0.

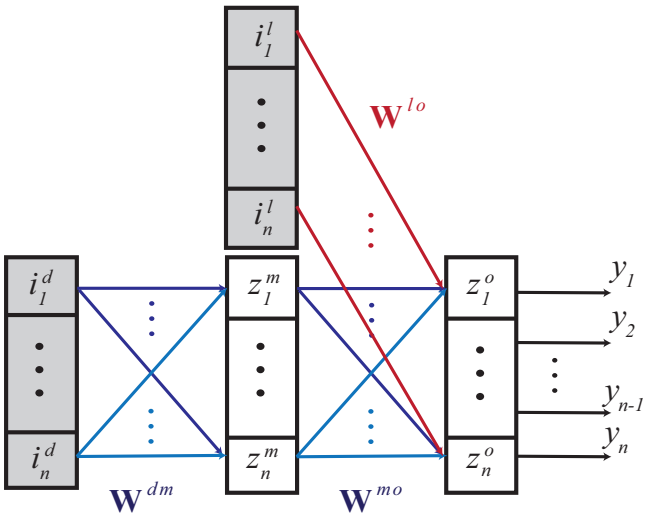

Fig. 2. The proposed design of the feed-forward NN for deflection routing. The input layer consists of two partitions denoted by binary vectors $\mathbf{I}^l = [i_1^l \ \dots \ i_n^l]$ and $\mathbf{I}^d = [i_d^l \ \dots \ i_n^d]$. The $\mathbf{I}^l$ partition of the input has weighted connections to the output layer. The binary vector $\mathbf{Z}^m = [z_1^m \ \dots \ z_n^m]$ denotes the mid-layer of the proposed feed-forward network while $\mathbf{Z}^o$ denotes the output layer.

The $\mathbf{I}^l$ partition of the input has weighted connections to the output layer. The $n \times n$ matrix of wights $\mathbf{W}^{lo}$ is defined as:

$$\mathbf{W}^{lo} = \begin{bmatrix} w_{11}^{lo} & 0 & \cdots & 0 \\ 0 & w_{22}^{lo} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & w_{nn}^{lo} \end{bmatrix}. \tag{8}$$

Selecting a link that is blocked for deflection should be avoided because it results in an immediate packet loss. Let us assume that the ith outgoing link of the buffer-less node is selected as the preferred link for deflection when the output y_i of the feed-forward network is 1. If Bernoulli-logistic agents are employed, selecting a blocked link for deflection may be prohibited by setting the elements w_{kk}^{lo}, $k = 1, \dots, n$, of the weight matrix $\mathbf{W}^{lo}$ to a large negative value. Regardless of the state of the buffer-less network and its behavior, selecting a blocked link will always result in immediate packet drop. This behavior does not change with time and does not depend on the reinforcement signals received from the environment. Hence, $\mathbf{W}^{lo}$ is a deterministic weight matrix that is not updated when reinforcement signals are received.

Let the binary vector $\mathbf{Z}^m = [z_1^m \ \dots \ z_n^m]$ denote the mid-layer of the proposed feed-forward network shown in Fig. 2. This layer is connected to the input and the output layers using the weight matrices:

$$\mathbf{W}^{dm} = \begin{bmatrix} w_{11}^{dm} & w_{12}^{dm} & \cdots & w_{1n}^{dm} \\ w_{21}^{dm} & w_{22}^{dm} & \cdots & w_{2n}^{dm} \\ \vdots & \vdots & & \vdots \\ w_{n1}^{dm} & w_{n2}^{dm} & \cdots & w_{nn}^{dm} \end{bmatrix} \tag{9}$$

and

$$\mathbf{W}^{mo} = \begin{bmatrix} w_{11}^{mo} & w_{12}^{mo} & \cdots & w_{1n}^{mo} \\ w_{21}^{mo} & w_{22}^{mo} & \cdots & w_{2n}^{mo} \\ \vdots & \vdots & & \vdots \\ w_{n1}^{mo} & w_{n2}^{mo} & \cdots & w_{nn}^{mo} \end{bmatrix}. \tag{10}$$

These matrices reflect the preferences for deflection decisions. If there is no initial preference for the output links, a uniform distribution of the output links is desirable. This is achieved by setting $\mathbf{W}^{dm} = \mathbf{W}^{mo} = \mathbf{0}$. However, these weight matrices get updated and new probability distributions are shaped as reinforcement signals are received from the environment. For an arbitrary $h \times k$ matrix $\mathbf{Q}$, we define the Bernoulli semilinear operator $\mathcal{F}$ as:

$$\mathcal{F}(\mathbf{Q}) = \begin{bmatrix} \frac{1}{1+e^{q_{11}}} & \cdots & \frac{1}{1+e^{q_{1k}}} \\ \frac{1}{1+e^{q_{21}}} & \cdots & \frac{1}{1+e^{q_{2k}}} \\ \vdots & & \vdots \\ \frac{1}{1+e^{q_{h1}}} & \cdots & \frac{1}{1+e^{q_{hk}}} \end{bmatrix}. \tag{11}$$

In each decision-making epoch, a probability vector $\mathbf{P}^m = [p_1^m \ \dots \ p_n^m]$ is calculated using the Bernoulli semilinear operator as:

$$\mathbf{P}^m = \mathcal{F}\left(\mathbf{I}^d \times \mathbf{W}^{mo}\right). \tag{12}$$

Each $z_k^m \in \mathbf{Z}^m$ is then calculated as:

$$z_k^m = \begin{cases} 0 & \text{if } p_k^m < 0.5 \\ \mathcal{U}(0,1) & \text{if } p_k^m = 0.5 \\ 1 & \text{if } p_k^m > 0.5 \end{cases}, \tag{13}$$

where $\mathcal{U}(0,1)$ denotes 0 or 1 sampled from a uniform distribution.

Let the $1 \times 2n$ row vector $\mathbf{I}^o = [\mathbf{I}^l \, \mathbf{Z}^m]$ denote the input to the output layer $\mathbf{Z}^o$. The $n \times 2n$ matrix $\mathbf{W}^o = [\mathbf{W}^{lo} \, \mathbf{W}^{mo}]$ denotes the weight matrix of the output layer $\mathbf{Z}^o$. The probability vector $\mathbf{P}^o = [p_1^o \ \dots \ p_n^o]$ is first calculated as:

$$\mathbf{P}^o = \mathcal{F}\left(\mathbf{I}^o \times \left(\mathbf{W}^o\right)^T\right). \tag{14}$$

We then calculate the binary output vector $\mathbf{Z}^o = [z_1^o \ \dots \ z_n^o]$ (13). Indices of 1s that appear in the output vector $\mathbf{Z}^o$ may be used to identify an outgoing link for the burst to be deflected. If $\mathbf{Z}^o$ contains multiple 1s, then the tie-break procedure described in Algorithm 1 is used for selecting the outgoing link. The output vector $\mathbf{Z}^o$ and associated probability vector $\mathbf{P}^o$ are the inputs to the algorithm. Multiple indices of 1s in $\mathbf{Z}^o$ may have different values in $\mathbf{P}^o$ because (13) maps to 1 all $\mathbf{P}^o$ elements greater than 0.5 when updating $\mathbf{Z}^o$. Therefore, in order to break a tie, Algorithm 1 considers all indices of 1s in $\mathbf{Z}^o$ and selects the index that has the maximum value in $\mathbf{P}^o$. If there are multiple indices with the same maximum value, Algorithm 1 randomly selects one of these indices according to a uniform distribution. It then sets to 0 the value of the elements of $\mathbf{Z}^o$ that were not selected. The algorithm needs to inspect the values of $\mathbf{Z}^o$ and $\mathbf{P}^o$ at most once. Furthermore, since vectors $\mathbf{Z}^o$ and $\mathbf{P}^o$ of a buffer-less node with n outgoing links have n elements, the time complexity of Algorithm 1 is $\mathcal{O}(n)$.

After the reinforcement signal is received from the environment, the network updates its weight matrices $\mathbf{W}^{dm}$ and $\mathbf{W}^{mo}$ according to (6).

Algorithm 1: The tie-break procedure used to select an action based on the output vector $\mathbf{Z}^o$ of the feed-forward NN.

Input: Output vector: $\mathbf{Z}^o = [z_1^o \ \ldots \ z_n^o]$
Probability vector: $\mathbf{P}^o = [p_1^o \ \ldots \ p_n^o]$
Output: Index of the outgoing link for the burst to be deflected: a

```
begin
    p_max ← 0
    S ← ∅
    for all k ∈ {1, 2, ..., n} do
        if z_k^o = 1 && p_k^o = p_max then
            S ← k
        end
        else if z_k^o = 1 && p_k^o > p_max then
            S ← ∅
            S ← k
            p_max ← p_k^o
        end
    end
    a ← sample uniformly from S
    for all k ∈ {1, 2, ..., n}\{a} do
        z_k^o ← 0
    end
    return a
end
```

C. Feed-Forward NN for Deflection Routing with k-Episode Updates

The difference between an episodic feed-forward NN decision-making module and a single episode module is that the episodic network may generate deflection decisions while it waits for reward signals that belong to its previous decisions. An episodic network generates deflection decisions similar to the feed-forward NN with a single episode. It keeps an episode counter $C_{episode}$ that is incremented when the network generates a deflection decision and a reward counter C_r that is incremented when the network receives a reward signal from the environment.

An episode starts when $C_{episode} = 0$ and terminates when the network receives a reward signal for all generated deflection decisions ($C_{episode} = C_r$). During an episode, the decision-making module maintains the input vector $\mathbf{I}^d$, weight matrices $\mathbf{Z}^m$ and $\mathbf{Z}^o$, and probability vectors $\mathbf{P}^m$ and $\mathbf{P}^o$.

Let us assume that an episode terminates after k decisions. Let $\mathcal{R}_k$ denote the sum of all k reward signals that have been received from the environment. When the k-episode terminates, the feed-forward network resets the counters $C_{episode}$ and C_r to zero. Furthermore, it updates its weights as:

$$w_{uv}^{dm} \leftarrow w_{uv}^{dm} + \left(\alpha \mathcal{R}_k \sum_{q=2}^{k} \left[\left((z_u^m)_q - (p_u^m)_q \right) (i_v^d)_{q-1} \right] \right) \quad (15)$$

$$w_{uv}^{mo} \leftarrow w_{uv}^{mo} + \left(\alpha \mathcal{R}_k \sum_{q=2}^{k} \left[\left((z_u^o)_q - (p_u^o)_q \right) (z_v^m)_{q-1} \right] \right), \quad (16)$$

where w_{uv}^{dm} and w_{uv}^{mo} are elements in the u^{th} row and v^{th} column of the weight matrices $\mathbf{W}^{dm}$ and $\mathbf{W}^{mo}$, respectively, and $(z_u^m)_q$ is the u^{th} element of the vector $\mathbf{Z}^m$ that was calculated during the q^{th} decision of the current episode.

D. Time Complexity Analysis

Q-NDD, PQDR, and RLDRS employ the table-based Q-learning algorithm. Hence, selecting an action depends on the table implementation. If the Q-table is implemented as a hash table, then an actions may be generated in constant time $\mathcal{O}(1)$. The update procedure executed after receiving a reward signal may also be completed in constant time. The NN-NDD and ENN-NDD algorithms employ the NN shown in Fig. 2. In the case of NN-NDD, time complexity of selecting an action and updating the network weights after receiving a reward signal is no longer constant. When selecting an action, the NN needs to calculate vectors $\mathbf{P}^m$ and $\mathbf{P}^o$ using (12) and (14), respectively. This results in $\mathcal{O}(n^2)$ complexity, where n is the number of a buffer-less node neighbors. The reward procedure requires inspection of elements of the weight matrices $\mathbf{W}^{dm}$ and $\mathbf{W}^{mo}$. Each of these inspections is quadratic in n, yielding a complexity of $\mathcal{O}(n^2)$ for the update procedure. In the case of ENN-NDD, the complexity of selecting an action is also $\mathcal{O}(n^2)$ while the complexity of the update procedure is $\mathcal{O}(kn^2)$, where k is the length of an episode.

The polynomial time complexity of the NN-NDD and ENN-NDD algorithms may affect their real-time decision-making performance. However, this complexity only depends on the degree of a buffer-less node. For example, the NN of a node with the degree equal to 1,000 performs 10^6 operations for an action selection or for an update. An average general-purpose 2 GHz CPU is capable of processing 2×10^9 operations per second and, therefore, it is capable of processing the NN in less than 0.5 ms.

VIII. Performance Evaluation

We evaluate performance of the proposed deflection routing protocols, RLDRS [9], and PQDR [10] by implementing them within the iDef framework. We compare the algorithms based on burst loss probability, number of deflections, average end-to-end delay, and average number of hops traveled by bursts. We first use the National Science Foundation (NSF) network topology shown in Fig. 3, which has been extensively used to evaluate performance of OBS networks [9], [18], [21], [30], [36]–[38]. We also use network topologies generated by the Waxman algorithm [39]. These networks consist of 10, 20, 50, 100, 200, 500, and 1,000 nodes. We compare the deflection routing algorithms in terms of memory and Central Processing Unit (CPU) time using the larger Waxman graphs. In all simulation scenarios, we allow up to two deflections per burst ($\text{DHC}_{max} = 2$). The burst header processing time is set to 0.1 ms.

A. NSF Network Scenario

The topology of the NSF network is shown in Fig. 3. The nodes are connected using bidirectional 1 Gbps fiber links with

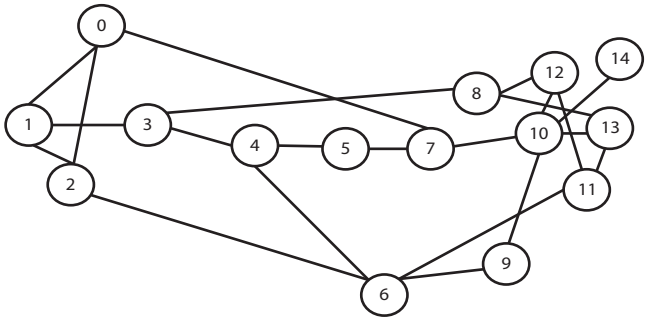

Fig. 3. Topology of the NSF network after the 1989 transition. Node 9 and node 14 were added in 1990.

64 wavelengths. The learning rate is set to $\alpha = 0.1$ and the maximum drop notification timer to $\mathrm{DN}_{max} = 50$ ms.

Multiple Poisson traffic flows with a data rate of 0.5 Gbps are transmitted randomly across the network. Each Poisson flow is 50 bursts long with each burst containing 12.5 kB of payload. While the burst arrival process depends on the aggregation algorithm [40] deployed in a node, the Poisson process has been widely used for performance analysis of OBS networks because it is mathematically tractable [41], [42].

The simulation scenarios are repeated with five random assignments of nodes as sources and destinations. The simulation results are averaged over the five simulation runs. The burst loss probability and the average number of deflections as functions of the number of Poisson flows for 64 wavelengths scenarios are shown in Fig. 4.

Even though the space complexity of the NDD algorithm is reduced to the degree of a node, simulation results show that NDD-based protocols perform better than RLDRS and PQDR in the case of low to moderate traffic loads. However, NN-NDD and Q-NDD protocols initiate larger number of deflections compared to RLDRS and PQDR. In moderate to high loads, PQDR exhibits the best performance. It deflects fewer bursts and its burst-loss probability is lower compared to other algorithms. In the cases of higher traffic loads, ENN-NDD algorithm discards fewer bursts while deflecting more bursts compared to other NDD-based algorithms.

In scenarios with lower traffic loads, the bursts are deflected less frequently and, therefore, the decision-making modules (learning agents) learn based on a smaller number of trials and errors (experience). The learning deficiency of Q-learning based algorithms in these cases results in higher burst-loss probabilities. In the cases of low to moderate traffic loads, the NN-NDD algorithm has the lowest burst-loss probability. In scenarios with higher traffic loads, decision-making modules (learning agents) deflect bursts more frequently. This enables the learning agents to gain additional experience and make more informed decisions. In these cases, RLDRS and the PQDR algorithm make optimal decisions, which result in lower burst-loss probabilities because they collect and store additional information about the network dynamics.

The RLDRS and PQDR signaling algorithms take into account the number of hops to destination when they generates feedback signals. Therefore, RLDRS and PQDR have smaller average end-to-end delay and average number of hops traveled by bursts. The average end-to-end delay of the deflection routing algorithms is shown in Fig. 5. The average number of hops has a similar trend.

Simulation results indicate that in the case of moderate loads (40%–65%), the NDD algorithms have much smaller burst-loss probability than RLDRS and PQDR, as shown in Fig. 4 (top left). For example, at 65% load, the burst-loss probability of the NN-NDD algorithm is approximately 0.003, which is four times better than performance of PQDR (≈ 0.012). The NDD algorithms maintain comparable performance in terms of end-to-end delay (within ≈ 0.04 ms), as shown in Fig. 5. In the case of high loads (80%–100%), the maximum difference in burst-loss probabilities is between NN-NDD (≈ 0.18) and PQDR (≈ 0.13) at 100% load, as shown in Fig. 4 (bottom left). The maximum difference in end-to-end delays at 100% load is between Q-NDD (≈ 0.5 ms) and RLDRS (≈ 0.42 ms), as shown in Fig. 5. Superior performance of NDD algorithms in case of moderate loads makes them a viable solution for deflection routing because the Internet backbone was engineered to keep link load levels below 50% [26]. Studies show that the overloads of over 50% occur less than 0.2% of a link life-time [26], [43].

B. Complex Network Topologies and Memory Usage

We use the Boston University Representative Internet Topology Generator (BRITE) [44] to generate random Waxman graphs [39] with 10, 20, 50, 100, 200, 500, and 1,000 nodes. An edge that connects nodes u and v exists with a probability:

$$\Pr\Big(\{u,v\}\Big) = \beta \exp\Big(\frac{-d(u,v)}{L\delta}\Big), \tag{17}$$

where $d(u,v)$ is the distance between nodes u and v, L is the maximum distance between two adjacent nodes, and β and δ are parameters in the range $(0,1]$. In simulation scenarios, we use $\beta = 0.2$ and $\delta = 0.15$ [45]. Nodes are randomly placed and each node is connected to three other nodes using bidirectional single wavelength fiber links. Sources and destinations of traffic flows are randomly selected. For all scenarios, we keep the network load at 40%. Hence, scenarios with 10, 20, 50, 100, 200, 500, and 1,000 nodes have 24, 48, 120, 240, 480, 1,200, and 2,400 Poisson traffic flows, respectively. Simulations were performed on a Dell Optiplex-790 with 16 GB memory and the Intel Core i7 2600 processor.

1) Burst-Loss Probability: Performance of deflection routing algorithms in terms of burst-loss probability as a function of number of nodes is shown in Fig. 6. Note that the burst-loss probability has a logarithmic trend. The NN-NDD and Q-NDD algorithms scale better as the size of the network grows.

The burst-loss probability of the NN-NDD and Q-NDD algorithms is smaller and bursts are deflected less frequently. However, bursts travel through additional hops and thus experience longer end-to-end delays. Therefore, smaller burst-loss probability and smaller number of deflections come at the cost of selecting longer paths, which are less likely to be congested. RLDRS and the PQDR algorithm consider the number of hops to destination when deflecting bursts, which causes the bursts to travel through shorter paths. The probability of congestion along shorter paths is usually higher because the majority of

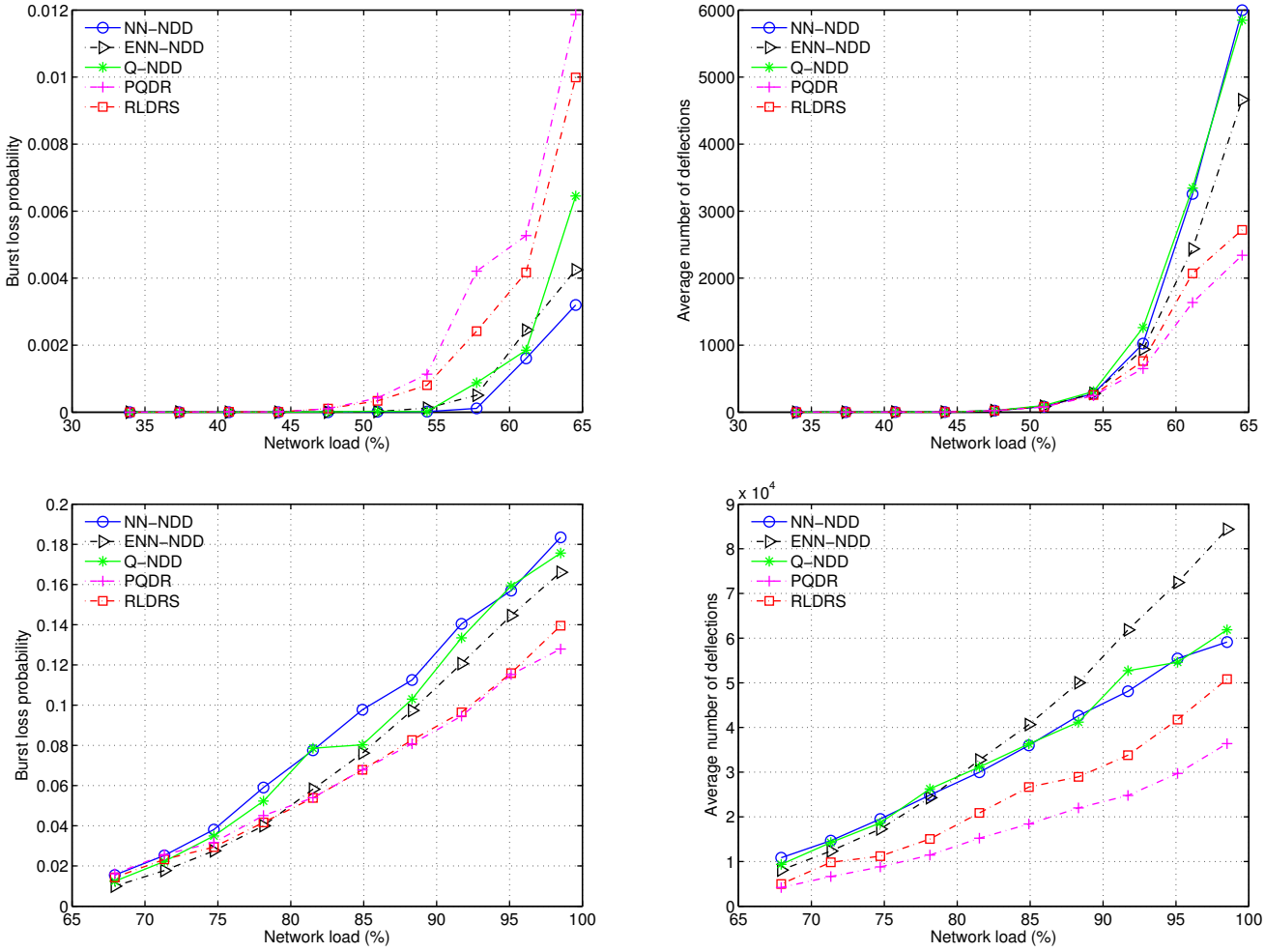

Fig. 4. Burst loss probability (left column) and average number of deflections (right column) as a function of the number of Poisson flows in the NSF network simulation scenario. For readability, two cases are plotted: 1,000 ($\approx 35\%$ load) to 2,000 ($\approx 65\%$ load) Poisson flows (top row) and 2,000 ($\approx 65\%$ load) to 3,000 ($\approx 100\%$ load) Poisson flows (bottom row). The NDD algorithms perform better than RLDRS and PQDR in case of low to moderate traffic loads. In the cases of higher traffic loads, ENN-NDD has smaller burst-loss while deflecting bursts more frequently compared to other NDD algorithms.

the routing protocols tend to route data through such paths. As a result, burst-loss probability and probability of deflecting bursts is higher along the paths that PQDR and RLDRS select for deflection.

2) Number of Deflections: Although burst deflection reduces the burst-loss probability, it introduces excess traffic load in the network. This behavior is undesired from the traffic engineering point of view. Therefore, the volume of the deflected traffic should also be considered as a performance measure. Performance of the deflection routing algorithms in terms of number of deflections as a function of number of nodes is shown in Fig. 7. Simulation results show that the NN-NDD and Q-NDD algorithms deflect fewer bursts compared to RLDRS and the PQDR algorithm.

3) End-to-End Delay and Average Number of Hops: Average end-to-end delay as a function of the number of nodes is shown in Fig. 8. Simulation results indicate that in the case of NDD algorithms, bursts travel through additional hops compared to RLDRS and the PQDR algorithm. When deflecting a burst, RLDRS and the PQDR algorithm consider

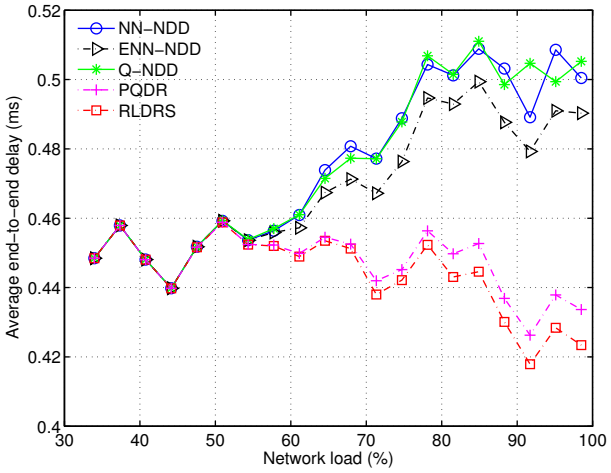

Fig. 5. Average end-to-end delay as a function of the number of Poisson flows in the NSF network scenario with 64 wavelengths. RLDRS and PQDR achieve better performance in terms of average end-to-end delay.

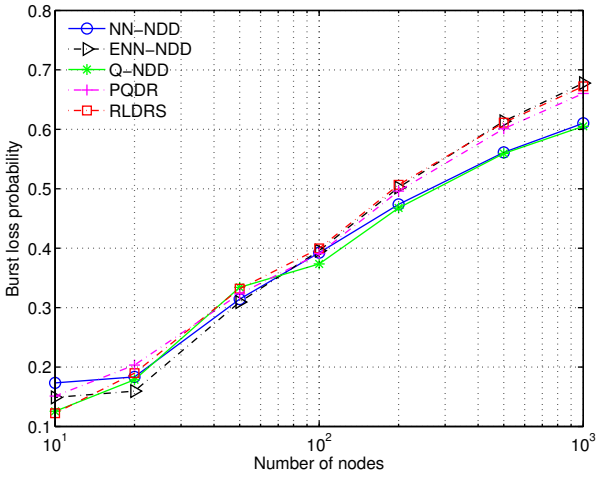

Fig. 6. Burst loss probability as a function of the number of nodes in the Waxman graphs at 40% traffic load.

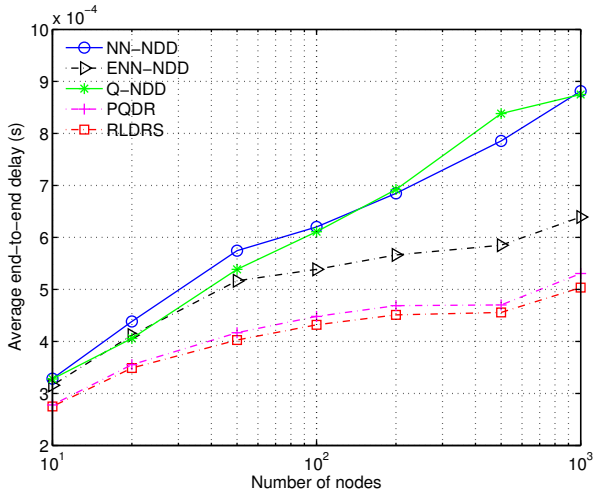

Fig. 8. Average end-to-end delay as a function of the number of nodes in the Waxman graphs at 40% traffic load.

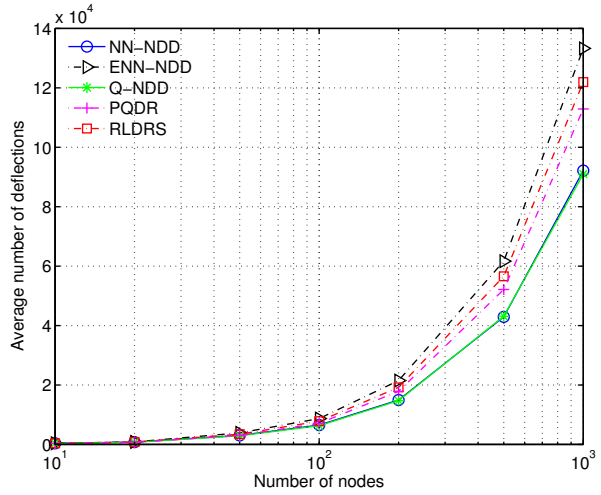

Fig. 7. Number of deflections as a function of the number of nodes in the Waxman graphs at 40% traffic load.

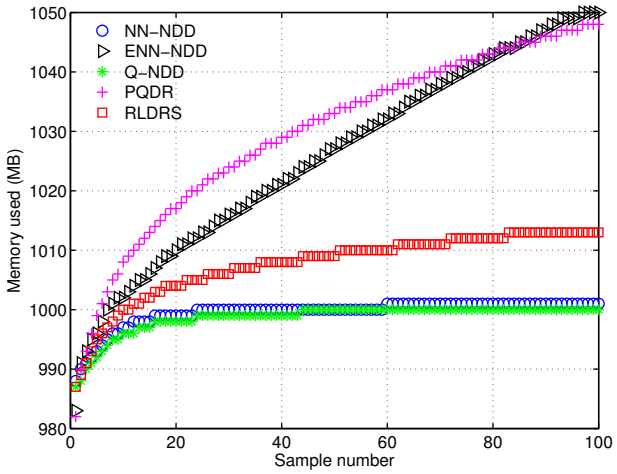

Fig. 9. Memory used in the network with 1,000 nodes. The graphs were generated by using 100 equally spaced time instances over each simulation run.

the number of hops to the destination. Furthermore, the underlying topology and the connectivity of nodes affect the number of hops traveled by bursts [23].

4) Memory and CPU Requirements: The memory and CPU time requirements of NN-NDD, ENN-NDD, Q-NDD, RLDRS, and PQDR are shown in Table I. All algorithms initially have comparable memory requirements. However, as the simulations proceed and the Q-tables are populated by new entries, the memory usage of RLDRS and the PQDR algorithm grows faster compared to NN-NDD and Q-NDD. The ENN-NDD algorithm memory usage grows faster than NN-NDD, Q-NDD, and RLDRS. This may be attributed to a larger number of bursts that ENN-NDD deflects, as shown in Fig. 7. The simulation results also show that NDD-based algorithms require less CPU time compared to RLDRS and PQDR. The memory usage of algorithms in the network with 1,000 nodes is shown in Fig. 9. The graphs were generated by using 100 equally spaced time instances over each simulation run.

IX. Conclusion

In this paper, we introduced the iDef framework that was implemented in the ns-3 network simulator. iDef may be employed for implementation and testing of various reinforcement learning-based deflection routing algorithms. Its independent modules enable users to integrate various learning and signaling algorithms when designing deflection routing protocols. We also introduced the NDD signaling algorithm for deflection routing. Its space complexity depends only on the node degree. Furthermore, we proposed an NN-based associative learning algorithm and an NN-based associative learning algorithm with episodic updates. We combined the NDD signaling algorithm with NN, ENN, and a Q-learning-based decision-making modules for deflection routing.

TABLE I
COMPARISON OF MEMORY AND CPU USAGE OF NN-NDD, ENN-NDD, Q-NDD, PQDR, AND RLDRS

Algorithm	Number of nodes	Number of links	Number of flows	Minimum memory usage (MB)	Maximum memory usage (MB)	Average CPU usage (%)	Total CPU time (mm:ss)
NN-NDD	500	1,500	1,200	264	271	8.1	8:28.23
	1,000	3,000	2,400	983	1,001	14.03	32:47.91
ENN-NDD	500	1,500	1,200	264	294	7.74	12:03.48
	1,000	3,000	2,400	989	1,050	12.69	42:40.16
Q-NDD	500	1,500	1,200	264	271	8.22	8:25.29
	1,000	3,000	2,400	987	1,000	13.56	32:35.44
PQDR	500	1,500	1,200	264	290	9.83	13:28.11
	1,000	3,000	2,400	987	1,048	15.93	52:58.90
RLDRS	500	1,500	1,200	264	276	9.53	14:27.04
	1,000	3,000	2,400	987	1,013	15.41	56:13.34

Performance of the NN-NDD, ENN-NDD, and Q-NDD algorithms was compared with the existing RLDRS and the PQDR algorithm. We implemented these algorithms within the iDef framework. For simulations, we employed the National Science Foundation (NSF) network topology and random graphs that consisted of 10, 20, 50, 100, 200, 500, and 1,000 nodes. These graphs were generated using the Waxman algorithm.

In simulations using the NSF network topology, NN-NDD achieves the lowest burst loss probability in the cases of low to moderate loads. Simulations with Waxman topologies indicate that NN-NDD and Q-NDD achieve smaller burst-loss probabilities while they deflect bursts less frequently. However, bursts travel through additional hops and thus experience longer end-to-end delays. Therefore, smaller burst-loss probability and smaller number of deflections come at the cost of selecting longer paths, which are less likely to be congested. RLDRS and the PQDR algorithm consider the number of hops to destination when deflecting bursts. This, in turn, causes the bursts to travel through shorter paths. However, the probability of congestion along shorter paths is usually higher because the majority of the routing protocols tend to route data through such paths. Consequently, burst-loss probability and probability of deflecting bursts are higher along the paths that RLDRS and the PQDR algorithm select for deflection. The proposed NDD signaling algorithm also requires less memory and CPU resources, which are more significant as the size of the network grows.

Most existing deflection routing algorithms have been evaluated using various simulation and analytical methods. They suggest that deflection routing is an effective approach to reduce burst-loss. However, experimental performance evaluation using testbeds remains an open research topic.

ACKNOWLEDGMENT

The authors thank W. W.-K. Thong and G. Chen from City University of Hong Kong and M. Arianezhad from Simon Fraser University for their contributions at the early stage of this project.

REFERENCES

[1] C. Qiao and M. Yoo, "Optical burst switching (OBS)—a new paradigm for an optical Internet," *J. of High Speed Netw.*, vol. 8, no. 1, pp. 69–84, Mar. 1999.

[2] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: a reinforcement learning approach," in *Advances in Neural Inform. Process. Syst.*, J. Jack, D. Cowan, G. Tesauro, and J. Alspector, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers, 1994, vol. 6, pp. 671–678.

[3] S. P. M. Choi and D. Y. Yeung, "Predictive Q-routing: a memory-based reinforcement learning approach to adaptive traffic control", in *Advances in Neural Inform. Process. Syst.*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. Cambridge, MA, USA: The MIT Press, 1996, vol. 8, pp. 945–951.

[4] L. Peshkin and V. Savova, "Reinforcement learning for adaptive routing," in *Proc. Int. Joint Conf. Neural Netw.*, Honolulu, HI, USA, May 2002, vol. 2, pp. 1825–1830.

[5] C. J. C. H. Watkins and P. Dayan, "Technical note, Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992.

[6] (2014, Oct.) The ns-3 network simulator [Online]. Available: http://www.nsnam.org/

[7] (2014, Oct.) iDef ns-3 implementation repository [Online]. Available: http://bitbucket.org/shaeri/hmm-deflection/

[8] S. Haeri, W. W-K. Thong, G. Chen, and Lj. Trajković, "A reinforcement learning-based algorithm for deflection routing in optical burst-switched networks," in *Proc. IEEE Int. Conf. Inf. Reuse and Integration*, San Francisco, USA, Aug. 2013, pp. 474–481.

[9] A. Belbekkouche, A. Hafid, and M. Gendreau, "Novel reinforcement learning-based approaches to reduce loss probability in buffer-less OBS networks," *Comput. Netw.*, vol. 53, no. 12, pp. 2091–2105, Aug. 2009.

[10] S. Haeri, M. Arianezhad, and Lj. Trajković, "A predictive Q-learning-based algorithm for deflection routing in buffer-less networks," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, Manchester, UK, Oct. 2013, pp. 764–769.

[11] H. G. Perros, *Connection-Oriented Networks: SONET/SDH, ATM, MPLS and Optical Networks*. Chichester, UK: John Wiley & Sons, 2005.

[12] Y. Xiong, M. Vandenhoute, and H. C. Cankaya, "Control architecture in optical burst-switched WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 1838–1851, Aug. 2000.

[13] A. Zalesky, H. Vu, Z. Rosberg, E. Wong, and M. Zukerman, "OBS contention resolution performance," *Perform. Eval.*, vol. 64, no. 4, pp. 357–373, May 2007.

[14] H.-L. Liu, B. Zhang, and S.-L. Shi, "A novel contention resolution scheme of hybrid shared wavelength conversion for optical packet switching," *J. Lightwave Technol.*, vol. 30, no. 2, pp. 222–228, Jan. 2012.

[15] X. Wang, X. Jiang, and A. Pattavina, "Efficient designs of optical LIFO buffer with switches and fiber delay lines," *IEEE Trans. Commun.*, vol. 59, no. 12, pp. 3430–3439, Dec. 2011.

[16] A. I. Abd El-Rahman, S. I. Rabia, and H. M. H. Shalaby, "MAC-layer performance enhancement using control packet buffering in optical burst-switched networks," *J. Lightwave Technol.*, vol. 30, no. 11, pp. 1578–1586, June 2012.
[17] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: a survey," *J. of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.
[18] S. Li, M. Wang, E. W. M. Wong, V. Abramov, and M. Zukerman, "Bounds of the overflow priority classification for blocking probability approximation in OBS networks," *J. Opt. Commun. Netw.*, vol. 5, no. 4, pp. 378–393, Apr. 2013.
[19] S. Bregni, A. Caruso, and A. Pattavina, "Buffering-deflection tradeoffs in optical burst switching," *Photon. Netw. Commun.*, vol. 20, no. 2, pp. 193–200, Aug. 2010.
[20] M. Levesque, H. Elbiaze, and W. Aly, "Adaptive threshold-based decision for efficient hybrid deflection and retransmission scheme in OBS networks," in *Proc. 13th Int. Conf. Optical Network Design and Modeling*, Braunschweig, Germany, Feb. 2009, pp. 55–60.
[21] E. W. M. Wong, J. Baliga, M. Zukerman, A. Zalesky, and G. Raskutti, "A new method for blocking probability wvaluation in OBS/OPS networks with deflection routing," *J. Lightwave Technol.*, vol. 27, no. 23, pp. 5335–5347 Dec. 2009.
[22] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509-512, Oct. 1999.
[23] S. Haeri and Lj. Trajković, "Deflection routing in complex networks," in *Proc. IEEE Int. Symp. Circuits and Systems*, Melbourne, Australia, June 2014, pp. 2217–2220.
[24] F. Borgonovo, "Deflection routing," in *Routing in Communications Networks*. New Jersey: Prentice-Hall, 1995, pp. 263–306.
[25] X. Yang and D. Wetherall, "Source selectable path diversity via routing deflections," in *Proc. ACM SIGCOMM*, New York, NY, USA, Oct. 2006, pp. 159–170.
[26] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot, "An approach to alleviate link overload as observed on an IP backbone," in *Proc. IEEE INFOCOM*, Stanford, CA, USA, Mar. 2003, vol. 1, pp. 406–416.
[27] W. W.-K. Thong and G. Chen, "Jittering performance of random deflection routing in packet networks," *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 3, pp. 616–624, Mar. 2013.
[28] J. Perelló, F. Agraz, S. Spadaro, J. Comellas, and G. Junyent, "Using updated neighbor state information for efficient contention avoidance in OBS networks," *Comput. Commun.*, vol. 33, no. 1, pp. 65–72, Jan. 2010.
[29] W. W.-K. Thong, G. Chen, and Lj. Trajković, "RED-f routing protocol for complex networks," in *Proc. IEEE Int. Symp. Circuits and Systems*, Seoul, Korea, May 2012, pp. 1644–1647.
[30] Y. Kiran, T. Venkatesh, and C. Murthy, "A reinforcement learning framework for path selection and wavelength selection in optical burst switched networks," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 9, pp. 18–26, Dec. 2007.
[31] X. Gao and M. Bassiouni, "Improving fairness with novel adaptive routing in optical burst-switched networks," *J. Lightw. Technol.*, vol. 27, no. 20, pp. 4480–4492, Oct. 2009.
[32] R. J. Williams and J. Peng, "Function optimization using connectionist reinforcement learning algorithms," *Connection Science*, vol. 3, no. 3, pp. 241–268, 1991.
[33] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, May 1992.
[34] G.-B. Huang and H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Trans. Neural Netw.*, vol. 9, no. 1, pp. 224–229, Jan. 1998.
[35] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Budapest, Hungary, July 2004, pp. 985–990.
[36] G. Wu, W. Dai, X. Li, and J. Chen, "A maximum-efficiency-first multi-path route selection strategy for optical burst switching networks," *Optik—Int. J. Light and Electron Optics*, vol. 125 no. 10, pp. 2229-2233, May 2014.
[37] A. Belbekkouche, A. Hafid, M. Tagmouti, and M. Gendreau, "Topology-aware wavelength partitioning for DWDM OBS networks: a novel approach for absolute QoS provisioning," *Computer Networks*, vol. 54, no. 18, pp. 3264–3279, Dec. 2010.
[38] B. G. Bathula and V. M. Vokkarane, "QoS-based manycasting over optical burst-switched (OBS) networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 271–283, Feb. 2010.
[39] B. M. Waxman, "Routing of multipoint connections," *IEEE J. Sel. Areas Commun.*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.
[40] X. Mountrouidou and H. Perros, "On the departure process of the burst aggregation algorithms in optical burst switching," *J. of Comput. Netw.*, vol. 53, no. 3, pp. 247–264, Feb. 2009.
[41] A. Zalesky, H. Vu, Z. Rosberg, E. W. M. Wong, and M. Zukerman, "Modelling and performance evaluation of optical burst switched networks with deflection routing and wavelength reservation," in *Proc. INFOCOM*, Hong Kong SAR, China, Mar. 2004, vol. 3, pp. 1864–1871.
[42] X. Yu, J. Li, X. Cao, Y. Chen, and C. Qiao, "Traffic statistics and performance evaluation in optical burst switched networks," *J. Lightw. Technol.*, vol. 22, no. 12, pp. 2722–2738, Dec. 2004.
[43] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Netw.*, vol. 17, no. 6, pp. 6–16, Dec. 2003.
[44] (2014, Oct.) BRITE [Online]. Available: http://www.cs.bu.edu/brite/
[45] E. W. Zegura, K. L. Calvert, and M. J. Donahoo "A quantitative comparison of graph-based models for Internet topology," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 770–783, Dec. 1997.

Soroush Haeri (S'11) received the undergraduate degree from Multimedia University, Cyberjaya, Malaysia. He is currently pursuing the Ph.D. degree from the School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada.

He is a member of the Communication Networks Laboratory, Simon Fraser University. He was a Software Engineer with Wavelet Solutions, Subang Jaya, Malaysia, from 2010 to 2011. In 2013, he was a developer of Android applications at Simon Fraser University Radio Station CJSF, Burnaby, BC, Canada. His current research interests include communication networks, applications of machine-learning algorithms to routing in computer networks, and scalable routing architectures.

Ljiljana Trajković (S'78–M'86–F'05) received the Dipl.Ing. degree from the University of Pristina, Pristina, Yugoslavia, in 1974, the M.Sc. degrees in electrical engineering and computer engineering, both from Syracuse University, Syracuse, NY, USA, in 1979 and 1981, respectively, and the Ph.D. degree in electrical engineering from the University of California, Los Angeles, Los Angeles, CA, USA, in 1986.

She is currently a Professor with the School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada. From 1995 to 1997, she was a National Science Foundation Visiting Professor with the Electrical Engineering and Computer Sciences Department, University of California, Berkeley, Berkeley, CA, USA. She was a Research Scientist with Bell Communications Research, Morristown, NJ, USA, from 1990 to 1997 and a Technical Staff Member with AT&T Bell Laboratories, Murray Hill, NJ, USA, from 1988 to 1990. Her current research interests include high-performance communication networks, control of communication systems, computer-aided circuit analysis and design, and theory of nonlinear circuits and dynamical systems.

Dr. Trajković serves as the President of the IEEE Systems, Man, and Cybernetics Society (2014–2015) and has served as the President-Elect (2013), the Vice President Publications (2010 to 2011 and 2012 to 2013), the Vice President Long-Range Planning and Finance (2008 to 2009), a member of the Board of Governors (2004 to 2006). She is a Chair of the IEEE Circuits and Systems Society Joint Chapter of the Vancouver/Victoria Sections. She was the President (2007) and a member of the Board of Governors of the IEEE Circuits and Systems Society (2001 to 2003 and 2004 to 2005), a Chair of the IEEE Technical Committee on Nonlinear Circuits and Systems (1998), a Technical Program Co-Chair of International Symposium on Circuits and Systems (ISCAS) 2005, a Technical Program Chair and a Vice General Co-Chair of ISCAS 2004, an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMSPART I (1993 to 1995 and 2004 to 2005), the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMSPART II (1999 to 2001 and 2002 to 2003), and the IEEE CIRCUITS AND SYSTEMS MAGAZINE (2001 to 2003), and a Distinguished Lecturer of the IEEE Circuits and Systems Society (2002 to 2003 and 2010 to 2011).