NASA Public Access

Author manuscript

IEEE Trans Cogn Commun Netw. Author manuscript; available in PMC 2019 November 08.

Published in final edited form as:

IEEE Trans Cogn Commun Netw. 2018 August 14; 4(4): . doi:10.1109/TCCN.2018.2865387.

Resource Selection in Cognitive Networks With Spiking Neural Networks

Ricardo Lent [Senior Member, IEEE]

Department of Engineering Technology, University of Houston, Houston, TX 77494 USA

Abstract

This paper explores the feasibility of a spiking neural network-based approach to cognitive networking, that is potentially suitable for low-power neuromorphic chips. We discuss the design of a cognitive network controller (CNC), which can dynamically optimize the selection of resources for recurrent network tasks, based on both its assigned objectives and observations of the actual performance achieved by each resource. We present a coding strategy for the action decisions based on the time-to-fire of spikes, a learning algorithm, and a regulation method to keep synapse strengths within an adequate range. To evaluate the proposed method, we apply the CNC to a challenged network environment using simulation. In this scenario, the CNC requires to optimize the average file transfer time over a multichannel space communication link, which is available only for a time window because of orbital dynamics. Compared to conventional methods, we show that the CNC achieves its objective for a broad range of offered loads. We examine the impact of key system factors that include learning and space protocol parameters. The proposed CNC potentially fosters the development of new cognitive networking applications.

Index Terms

Networks; neural networks; learning systems; simulation; space vehicle communication

I. Introduction

Cognitive networking provides the means for devices to independently recognize and adapt to changes in the network environment or user demand. Such property proves to be beneficial to many current applications that rely on a shared use of network resources. The continuous advance of the computer and communication industry creates a situation where, coupled with a growing user base, a mix of new and legacy technologies operate together. This leads to a complex system of a massive number of heterogeneous elements of significant service performance variability. In many cases, the operational environment behaves erratically, for example, consider the case of challenged networks that include sparse wireless sensor networks and interplanetary networks where links are only sporadically available. These developments suggest a highly changing and mobile future scenario, redundant, and of time-varying performance, that requires efficient ways for the network to autonomously adapt to those changes for the benefit of the user. Much of the

progress in the field has been linked to physical and link layer problems (cognitive radio), e.g., for the dynamic selection of transmission rates, modulation, and the opportunistic use of the spectrum, but the scope of cognitive networking is much wider as suggested by this work and spans all layers of the protocol stack.

Through dynamic adaptation of operational parameters and actions, cognitive networking seeks to optimize performance metrics of interest. It defines autonomous processes and use machine learning to become aware of current network conditions through the partial, and often imprecise, observation of network metrics and service delivery performance. The implementation of autonomous decision-making processes seeks to select and execute specific resource management actions, exploiting the available knowledge of the network state, pre-assigned rules, internal state, and possibly, model predictions. Different types of actions are possible, which need to be customized based on specific application needs.

This paper is motivated by the need of efficient and effective learning approaches for cognitive networks to achieve the high performance that users expect. Our proposed method uses a spiking neural network (SNN) model to define a *Cognitive Network Controller* (CNC), which we show is able to achieve approximated self-awareness and a self-optimizing behavior in the recurrent selection of network resources for its assigned task. SNNs, which are generally identified as the third generation of neural networks, further evolve McCulloch-Pitts' and activation function-based neural networks [1] with a biologically accurate model of the synaptic interactions that occur in natural neurons. A spiking neuron model describes the analog–digital, temporal dynamics of neurons, unlike prior models that only describe their continuous behavior, which enables conveying a high information carrying capacity [2].

The main goal of this paper is to explore the feasibility of a biologically-inspired SNN approach to cognitive networking, motivated by the computational possibilities of a fast signal decoding that could be achieved with spiking neuron models through a time-based or rate-based decision coding. Moreover, the rapid development of neuromorphic chips provides suitable energy efficient and high-throughput hardware to run SNNs, expanding the practical application scope of this work [3]. The key contributions of this paper are:

1. Introduces a recurrent neural network of spiking neurons that is suitable for the cognitive selection of network resources (actions). The SNN codes action decisions in the timing of spikes. Prior works have explored the use of different neural network types, but not SNNs.
2. Presents a learning algorithm for this SNN, which modifies presynapse strengths according to observed network performance and with respect to a given objective function. We discuss how to find a suitable range for synaptic strengths and how to keep them within that range.
3. Shows the feasibility of the SNN approach using a realistic simulation of a multi-channel space communication scenario where the stations can only communicate directly for a short time window. We evaluate the impact of communication protocol parameters and learning parameters to give indication of the best values.

The rest of the paper is organized as follows. In Section II, we present a brief overview of the key aspects of SNNs for this study and use these concepts to design the CNC in Section III. To evaluate the effectiveness of the CNC, we selected a space communication scenario (Section IV). Simulation results can be found in Section V. A survey of the most relevant works in the literature appears in Section VI. Finally, we conclude the paper by proving a brief summary of the main results and future research directions in Section VII.

II. Spiking Neuron Model

We give a brief explanation of the spiking neuron model that we use to define our cognitive controller. For further details, we refer the reader to the specialized literature on the subject [4]. The model describes the neural dynamics with reasonable accuracy and at low complexity, which makes it attractive for our purposes. Biological neurons communicate through signals in the form of spikes. The spike generation and transfer from the emitting neuron (presynapse) to the receiving neuron (postsynapse) can be described by phenomenological models. One such model is the Leaky-Integrate-and-Fire (LIF), which explains the electrical properties of a neuron as consisting of a capacitor C connected in parallel with a resistor R . Fig. 1 depicts the LIF model. The circuit is driven by a current $I(t)$ and the voltage across these components gives the estimation of the neural membrane potential $u(t)$:

$$\tau \frac{d}{dt} u(t) = -u(t) + RI(t) \quad (1)$$

where τ is the time constant of the circuit ($\tau = RC$). The membrane potential gives the state of the neuron at time t (see right-hand side of Fig. 1).

A neuron may receive signals from other neurons or from an external source. The driver current $I(t)$ that stimulates a neuron is therefore, the aggregated effect of external stimuli, $i_e(t)$, and the trains of spikes generated at other neurons j that reach neuron i . We note that multiple synapses may connect any two neurons, possibly with different strengths and delays due to length differences in the connections. This observation leads to the following expression for the driver current:

$$I(t) = i_e(t) + \sum_f \sum_j \sum_k w_{jk} i_{jk}^{(f)}(t) \quad (2)$$

Each synapse is characterized by parameter w_{jk} , $w_{jk} \geq 0$, which gives the strength of the k -th synapse originated at neuron j . The spikes generated by the presynaptic neuron j produce a current $i_{jk}^{(f)}(t)$ on the postsynaptic neuron: $i_{jk}^{(f)}(t) = \delta(t - t_j^{(f)} - d_{jk})$, that is, it produces unit impulses that occur at the firing times f of the presynaptic neuron, but delayed by a time d_{jk} . A spike produces either an excitatory or inhibitory effect on the postsynapse, that is, it either increments or decrements the driver current $I(t)$. The kind of effect is determined by the type of presynapse neuron—either excitatory or inhibitory. In terms of postsynaptic potential change, the f -th spike produces the change $\epsilon(t - t_j^{(f)} - d_{jk})$, where $\epsilon(t) = \frac{t}{\tau} e^{1-t/\tau}$ with a preceding sign that depends on the kind of presynaptic neuron generating the spike. The

circuit on the left-hand side of Fig. 1 represents a synapse. A spike generated at the presynapse neuron arrives at the postsynapse after traversing a low-pass filter.

The analog description of the dynamics of a LIF unit changes drastically after its membrane potential reaches a certain threshold θ (with typical value -50 mV). After reaching θ , the neuron fires a spike that travels along its axon and then to other neurons through the synapses producing the analog-digital process on the other neurons. These spikes occur at firing times t^f : $u(t^f) = \theta$. After each spike, the membrane potential is reset to the value $u_r < \theta$ and remains at that value for a refractory period before resuming the analog dynamics. We observe that for any neuron, the next time-to-fire a spike depends on the aggregated current $I(t)$ and the current membrane potential $u(t)$. By adjusting the stimulus, it is possible to either accelerate, defer, or inhibit the next spike firing time. We use this observation to associate specific cognitive network actions to the neurons' time-to-fire.

III. SNN Controller

The goal of the Cognitive Network Controller (CNC) is to learn how to recurrently select network resources (actions) while optimizing a given objective when multiple redundant options are available for the same task, possibly operating at a different state and with different performance and efficiency levels. In general terms, the basic problem consists in the optimal selection of action a from a finite candidate set A for a cognitive networking task. The choice of the candidate set is application specific and to give a concrete example, in our evaluation scenario, the candidate set consists of radio frequency (RF) channels of different characteristics. The task of the CNC is to dynamically choose a channel for different file transfers (see Section IV). Other applications are certainly possible. The CNC relies on learning, which implies that it dynamically improves decisions based on its prior experience as it becomes aware of the environment. To support the operation of the CNC, we introduce a recurrent Spiking Neural Network (SNN) where neurons and synaptic connections behave according to (1) and (2).

The key idea is to link the activity of selected neurons ("core" neurons) to a cognitive action selection. The size of the action candidate set determines the size of the SNN. Specifically, for a problem of n possible actions, the SNN consists of the graph $G = (V, W)$, where $V = \{c_1, \dots, c_n, g, e_1, \dots, e_n, i_1, \dots, i_n\}$ and W is the set of synapses. Vertices $c_1, \dots, c_n$ form a core network of n recurrently connected excitatory neurons. Excluding self-connections, these core neurons are connected with non-negative weights $w(c_i, c_j) \in W, i, j = 1, \dots, n$. Inhibitory neurons provide the regulation required to maintain the network within a suitable operational range. For efficiency purposes, this is achieved by introducing a single inhibitory neuron g with multi-terminal postsynaptic connections to all core neurons with weights $w^k(g, c_i), i, k = 1, \dots, n$. Optionally, each core neuron i has additional multi-terminal postsynaptic connections from one presynaptic excitatory neuron e_i and one presynaptic inhibitory neuron i_i : $w^k(e_i, c_i), w^k(i_i, c_i), i, k = 1, \dots, n$. These neurons can be excited only by an external signal, E_i and I_i respectively, to either "encourage" or "discourage" specific actions. For example, as dictated by a schedule or by knowledge of the exclusion of a given asset, the corresponding action can be temporarily removed by inhibiting the related core neuron c_i through the generation of spikes from i_i .

In this SNN learning occurs by modifying according to some rule the weights $w(c_i, c_j) \in W$, $i, j = 1, \dots, n$, that is, the core neuron-to-core neuron synapses, and $w^k(g, c_i)$, $i, k = 1, \dots, n$ —the synapses from inhibitory neuron g . The rest of the synapses remain fixed. Also, we define a constant stimuli i_{ref} that is applied to all core neurons and neuron g , which serves to start the analog dynamics of the SNN.

We note that the number of vertices of the proposed SNN design including the optional connections is $|V| = 3n + 1$ and the number of edges is $|W| = n(n + 2)$. For example, Fig. 2 depicts the SNN for an 8-action CNC. The core excitatory neurons $c1$ – $c8$ are recurrently connected, with separate synapses each way, and neuron g introduces inhibitory regulation to the core neurons. The neurons receive a constant stimulus I_{ref} , which makes them emit at least the first spike. Optional neurons $e1$ – $e8$ and $i1$ – $i8$ provide further control over the spike generation dynamics. In addition to the SNN topology, we need to define two mechanisms to make this design suitable for a cognitive networking task: an action encoding and a learning rule.

A. Time-Based Decision Coding

The reason for constructing the SNN based on the size of the candidate set is to be able to associate each core neuron to each of the available actions. We define the CNC decision as the action that is associated to the core neuron emitting the earliest second spike. The first spike helps to “bootstrap” the SNN dynamics as follows. At time $t = 0$, all core neurons receive an initial stimulus in the form of a constant external current $i_e(t) = K$. The value of K is set sufficiently high to make each core neuron emit a spike at about the same time according to equation (1), unless an external inhibition signal $i_i(t)$ or external excitation signal is also applied.

Because of the recurrent structure of the SNN, emitted spikes will reach other core neurons after being modulated by the synaptic strengths as described by equation (2). This will cause neurons to continue firing at different times. Assuming that the synapse strengths were previously customized to reflect a desired action-selection performance goal, the core neurons will emit their second spike following a time order that reflects their relevance to the given performance goal. Specifically, the next action decision $a = i^*$ is provided by the earliest neuron firing for the second time:

$$c_{i^*} = \underset{c_i}{\operatorname{argmin}} t^{(2)}(c_i); i = 1, \dots, n \quad (3)$$

where $t^{(f)}(x)$ is the f -th time to fire of neuron x . As with other learning algorithms, it is possible to reach local minima. To prevent the issue, we superpose a random walk to this decision process, which occurs with probability P_w , that is, with probability P_w an action is selected randomly rather than by (3). Higher values of P_w allow better exploration of the network state, but may also lead to the selection of suboptimal actions. In contrast, lower values of P_w favors action optimally, but may lead to local minima. The value of P_w , therefore, controls the exploration and exploitation tradeoff of the learning process of the controller. In general, the value of P_w should be small (e.g., less than 0.2), but it is difficult

to define a specific value for P_w that works well for all cases. We later explore the impact of P_w to network performance in our evaluation scenario (Section IV).

B. Learning

The general idea is to stimulate spike firing activity at the neurons associated to the actions that offer the best performance and to inhibit the rest. Therefore, learning occurs by modifying the core and global inhibition weights of the SNN according to the observed performance of prior actions. One difficulty to address is that no fixed performance reference can be safely assumed for all cases given that such value would depend not only on the type of application, but also on time-varying conditions. A simple solution is to assume that the performance reference is the average of prior outcomes. The advantage of this approach is that it can adapt to both improving or degrading environment performance.

Each performance feedback from the system (for action i) can be associated to a certain cost C . Potentially, C can be a multivariate function of metrics of interest. The new average objective G is calculated from the current cost and the prior average: $G \leftarrow \alpha C + (1 - \alpha) G$, for $0 \leq \alpha \leq 1$. The learning rule changes the weights by an amount that is proportional to $\delta = G - C$:

$$\begin{aligned} w(c_j, c_i) &\leftarrow w(c_j, c_i) + \eta \delta; j = 1, \dots, n; i \neq j \\ w^k(g, c_i) &\leftarrow w^k(g, c_i) - \eta \delta; k = 1, \dots, n \end{aligned} \quad (4)$$

where $\eta > 0$ is the learning rate. The value of η can have a significant impact to network performance given that it controls how fast SNN weights change with respect to the new network state. Similarly, parameter α controls the short-term memory of the CNC. Higher values of α applies higher importance to the most recent performance observations, whereas lower values of α improve the importance of the feedback history. The selection of η and α is scenario specific and they would normally be determined by trial and error. As with P_{rw} , we evaluate the impact of selecting η and α in our evaluation scenario in Section IV.

We note that the complexity of the learning algorithm is $O(n)$, where n is the SNN size (number of actions). Because no dependencies exist between weight updates, it is possible to implement learning in parallel with a time complexity $O(1)$.

C. Synapse Strength Re-Scaling

By adapting the weights of the SNN according to (4), it is possible to either defer or accelerate the spike firings of any of the core neurons. However, this learning rule may also reduce or amplify certain weights to unsuitable levels. Weights that are too small may prevent some core neurons from firing. Likewise, neurons can saturate, and very large weights may produce no modification in the spike firings. The case is illustrated in Fig. 3 for a 2-neuron SNN connected through a single-terminal synapse. The presynaptic neuron was excited with a constant current of 2.1 nA and the synaptic weight was varied, at a 1-unit resolution, to observe the firing times of the postsynaptic unit.

The postsynaptic neuron did not fire for weights below 96. Weights above 600 produced no appreciable reduction of the time-to-fire. To prevent unsuitable weights, we suggest forcing the values to be in the range $[W_{min}, W_{max}]$ after applying the learning rule with:

$$\begin{aligned} w(c_i, c_j) &\leftarrow \gamma(w(c_i, c_j) - W_{min}) + W_{min} \\ w^k(g, c_i) &\leftarrow \gamma(w^k(g, c_i) - W_{min}) + W_{min} \end{aligned} \quad (5)$$

where W_{min} is the minimum allowable weight for any synapse of the SNN and $0 < \gamma < 1$ ($\gamma = 0.8$ in our tests). The maximum weight is denoted W_{max} . The values of W_{min} and W_{max} need to be determined beforehand. A very simple procedure to find the minimum and maximum weight values is given by Algorithm 1. Note that this process is only needed once for any SNN. Approximated values for W_{min} and W_{max} are enough. Algorithm 1 needs an initial estimation of a mid-level weight (within the bounds of the maximum and minimum weights) that needs to be experimentally determined. A value of 100 worked well with networks of size 2–100 in our tests. To minimize the processing overhead, the re-scaling process (5) should be restricted to the case when $(w_{max} - w_{min}) > \gamma(W_{max} - W_{min})$, where w_{max} and w_{min} the largest and smallest weights in the current SNN. After the calibration, it is straightforward to fix initial weights to the SNN (e.g., random in $[W_{min}, W_{max}]$). Note that as a result of the calibration and learning process, synapses' weights may be set to levels that may be biologically unrealistic. Also, spikes from a single presynapse could be sufficiently strong to cause an action potential on the postsynapse. It is possible to increase the biological realism of the design to address these observations. It will imply increasing the number of neurons and reducing the range of synaptic strengths. However, we could not identify evident benefits of improving the realism to the effectiveness of the CNC, which remains biologically-inspired rather than biologically accurate.

IV. Evaluation Scenario

The International Space Station (ISS) is a low-orbit artificial satellite that travels at 7.57 km/s at an average distance of 400 km above the surface of the Earth. Its orbit is almost circular with a perigee of 401 km and apogee of 408 km, but with a short period of revolution of 92.65 minutes on a 51.64-degree inclination. These orbital features limit direct communication opportunities between the ISS and a ground (Earth) station to 3 to 6 mins. Because of the short contact opportunity, the use of multiple links to concurrently transmit data between the ISS and a ground station can be of significant interest to move large data sets, in particular, for client-pull applications that involve certain level of interactivity. Such application is possible with low orbits because of the short propagation delays—of about 6 ms for the ISS. The assumption is therefore that the client (ground station) periodically requests data files (bundles) from the spacecraft while the link is available. In contrast, the use of one or more geostationary satellites acting as relays between a ground station and the ISS to extend the contact opportunity window can easily increase the total one-way propagation delay at least by a factor of 5, which is undesirable for this application.

Algorithm 1 SNN Initial Weight Calibration

```

1: procedure CALIBRATION( $SNN, w_{def}$ )
2:    $W_{min} \leftarrow w \leftarrow Fw_{def}$   $\triangleright F > 1$ 
3:   while True do
4:     Set all SNN weights to  $w$ 
5:     if all  $c_i$  and  $g$  neurons not fire at least twice then
6:       Exit loop
7:     end if
8:      $W_{min} \leftarrow w, w \leftarrow H w$   $\triangleright 0 < H < 1$ 
9:   end while
10:   $W_{max} \leftarrow w \leftarrow W_{min}$ 
11:  while True do
12:    Set all SNN weights to  $w$ 
13:    if average 2nd time-to-fire unchanged then
14:      Exit loop
15:    end if
16:     $W_{max} \leftarrow w, w \leftarrow w/H$ 
17:  end while
18:  return  $W_{min}, W_{max}$ 
19: end procedure

```

We use the ground station–ISS communication context to define an application scenario for the proposed CNC. As service-level objective (SLO), the aim is to achieve response times that are less than a second on average. To evaluate the scenario, we have developed an event-driven simulator that models the most relevant aspects of the real system to the communication, which include Earth–ISS orbital mechanics, radio propagation, bit transmission errors, packet buffering, and specific space protocols: the Licklider Transmission Protocol (LTP) and the Bundle Protocol (BP). Packet buffering follows the common first-come-first-served policy with no priority queuing. We further define that the ground station is located in Houston and that the ISS passes over that area offering a 5-minute communication window. This is assuming a minimum elevation of about 40 degrees to remove the impact of ground-level obstacles and the atmospheric effects near the horizon. Fig. 4 (a) depicts the simulated ISS orbit with its true anomaly corresponding to 03/23/2018 at about 6 pm central time and the location of the ground station. The distance of the ground station to the ISS for this contact period, as predicted by the simulator, is shown in Fig. 4 (b). The distance separating the stations during the ISS pass sets the signal propagation delay and the bit error rate for each packet transmission in the simulation.

The CNC consists of a SNN of 13 neurons. A summary of the SNN model parameters appears in Table I. The interest is in evaluating the ability of the CNC to learn the best channel for each file transfer while exploiting their performance differences to achieve low response times. Because of the buffering and transmission delays, in addition to the time-varying link quality, the best channel is likely different for every new file transmission. We consider four bi-directional channels that operate with carriers of 9, 11, 14, and 16 GHz. The former two channels are in the X-band and the latter two in the Ku-band. We use phase-shift keying (PSK) in all channels with a common transmission power of -3 dB, total antenna gains of 70 dB, and coding gain of 6 dB. The transmission rate of the X-band channels was set to 256 Kbps each. However, the LTP protocol of these channels was configured with a different number of sessions—set to 5 and 10 respectively. This parameter sets the maximum number of concurrent session transmissions, allowing the second channel to achieve twice as

much throughput for the same load as long as it does not exceed the link capacity. The number of sessions of LTP in the Ku-band channels was also set to 10, but with transmission rates of 512 Kbps and 2 Mbps respectively. The distance between the stations was used to find the timeout intervals for each new LTP session and the BER value that determines the maximum number of LTP rounds allowed by the protocol before aborting a block transmission. BP, LTP, and channel parameters are usually fixed by mission control and the role of the CNC in this scenario is to exploit the existing resources, as they are provided, to achieve its assigned objective.

Each simulation run consists in the generation of random requests to the ISS from the ground station at rate λ with an exponential interarrival distribution. Each request attempts to retrieve a file of average size L from the ISS. The model for file sizes was also exponentially distributed. The choice of an exponential traffic model conveniently allows the verification of simulator outputs with well-known theoretical results, at least with static routing probabilities. We also verified the performance of the space protocols (BP and LTP) using theoretical models and experimental studies from the literature.

The size of the request is fixed (100 bytes). Requests travel in bundles as data files do. We removed the bundle aggregation threshold and time limit constraints so that small bundles (i.e., requests) can be sent out without a deferring time. Each file is carried by a separate bundle and each bundle fits within a single 100% red-block LTP block for reliable delivery. LTP blocks are then sent as a number of segments, whose value depends on the maximum segment size that is a factor in the experiments. Rejected segments are sent by LTP up to a predetermined number of times and if such number is exceeded, the block is discarded. These features realistically follow the behavior of the real space protocols.

V. Results

For performance reference, we have included results that we obtained using three basic, non-cognitive selection strategies: round robin, random walk, and reflex (greedy action). Since our main goal is to introduce and demonstrate the feasibility of the CNC–SNN concept, we believe that these basic methods provide good reference performance as they are widely used either as such or as part of more elaborate methods. Round robin is commonly used by various types of communication applications (e.g., time-division multiple access–TDMA). It deterministically rotates the output channel for each new packet transmission. A similar situation occurs with random walk, but with a selection that follows a uniform probability distribution function. Both round robin and random walk attempt to distribute equally the incoming traffic among the available resources, without consideration of the actual channel performance, and have constant time complexity. In contrast, the reflex action method selects the channel of the best observed performance. This is achieved by maintaining the vector $\{d_i\}$, for $i = 1, \dots, n$, where d_i is the metric of interest associated to channel i . The method has constant time complexity with hashing or linear if using a simple list search.

In this work, the metric of interest is the file transfer time, which is extracted from the file request–response times. After completing a file transfer over channel i , the vector is updated with: $d_i \leftarrow \alpha_0 r_k + (1 - \alpha_0) d_i$, where r_k is the response time the k -th file. The reflex action is

simply obtained by selecting $\text{argmin}_i \{d_i\}$. We only consider the pure reflex method: $\alpha_0 = 1$ to avoid introducing an extra factor to the evaluation. The study focuses on observing the impact of the user demand (offered load λ), protocol configuration (LTP segment length m), and CNC learning parameters (α , η , and P_{rw}). When possible, we indicate the 95% confidence interval of the mean in the charts that follow. The number of samples (complete simulation runs) for each average value reported in the figures the follow varies between 300 and 8000 and were obtained with the aid of a high-performance computer cluster.

A. Impact of the Offered Load

The response time of each file is the time elapsed between the request and the file arrivals both at the ground station. It includes buffering time, propagation delay, and the time spent on the transmission and all retransmissions. User demand intensity was modeled with exponential interarrival times with mean λ^{-1} . Fig. 5 depicts the response time with LTP segment lengths of 1000 or 2000 bytes vs. the offered load λ .

We note that the packet loss ratio nearly duplicates moving from 1000-byte to 2000-byte segments as the packet loss probability is given by $p = 1 - (1 - p_e)^L$, where p_e is the channel's bit error rate and L is the packet length (bits). The higher loss rates impact performance due to the extra retransmission attempts that LTP must handle. We observed a reduction of about 20% in the maximum effective load to maintain the average response time below one second.

The CNC produced lower average response times overall than the other methods except under the heaviest load near the 1-second response time mark. Compared to the reflex method, the CNC was able to deliver files with lower delay at low and mid load levels than all of the other methods. Beyond the usable range depicted in Fig. 5, the reflex method tends to perform better than the CNC. A likely explanation for this effect is that the extended buffering times that occur under high demand create excessive delay to the provision of up-to-date information for CNC learning. This causes the CNC to learn stale information and as a result, to produce suboptimal actions. Our future work explores solutions to this issue, possibly by replacing the feedback with a mechanism that is not sensitive to congestion, or by switching methods for the high demand region. Both random walk and round robin, which do not consider channel performance, produced the worst performance as anticipated. We observed that congestion builds up quickly with these two methods and reduce the usable range to less than half compared to reflex or CNC.

In terms of throughput, the CNC and the reflex method yielded similar outcomes followed by round robin and random walk (see Fig. 6). Throughput was not directly formulated into the learning goal of the CNC, but it is interesting to note that despite producing lower delay than the other methods, the CNC did not add any throughput penalty.

B. Impact of the Segment Length

The choice of the segment length directly impacts the probability of packet loss as mentioned in the previous section. For each packet transmission, the simulator determines the distance between the stations to calculate the carrier-to-noise and the energy per bit to noise power spectral density ratio (E_b/N_0) to then find the bit error rate for the transmission

p_e using well known results for PSK modulation: $p_e = \text{erfc}(0.5E_b/N_0)/2$. A direct implication of changing the segment length is that larger segments become more prone to errors, but LTP sessions also become smaller. In LTP, the transmission of each data block proceeds as the transmission of a sequence of segments. LTP uses a negative acknowledgment strategy to retransmit the segments lost in transit. During the first transmission round, LTP transmits the entire block and labels the last segment as the checkpoint segment (CS). On reception of the CS, the receiver generates a report to the sender that serves as a request for the retransmission of the lost segments during the round. The process repeats for a number of rounds until all of the segments from the block are delivered, or after reaching the certain maximum number of attempts. In the latter case, the block is discarded. Block transmissions can occur in parallel with a user configuration limit. The segment length is also user configurable. Both parameters, the number of sessions (maximum parallel block transmissions) and the maximum segment size can be set by the user in the Interplanetary Overlay Network reference implementation of LTP [5]. These values can have a large impact on performance. We explore the impact by varying the segment length for values of the offered load of 1, 3, and 5 req/s and fixing a different number of sessions for each channel. The values for the offered load cover most of the usable range that produces acceptable response times (less than a second) for this application.

Fig. 7 depicts the results obtained with the CNC and the three reference methods: (a)–(c) show the response time for offered loads of 1, 4 and 7 req/s respectively, and (d) depicts the observed throughput. As expected, random walk and round robin yielded the longest response times. We observed that the performance of the reflex method was quite sensitive to the choice of segment length and the corresponding packet loss ratio across all values of λ . Similar sensitivity was observed for the CNC, but only for the highest workload level. We note that at this level, the outcome of round robin and random walk was well above 1 second, so those results are not visible in Fig. 7(c). In terms of throughput, the best performing methods, CNC and reflex, produced about the same outputs under different segment lengths.

C. Impact of the Learning Parameters

The three most relevant parameters that yield the learning behavior of the CNC are α , η , and P_{rw} . Parameter α determines the short-term memory of the controller. It provides a weight for the most recent sample in the calculation of the average response time of the system. Parameter η is the learning speed factor, which amplifies or reduces the impact of the performance difference between the selected action and the anticipated performance when updating the SNN weights. To enable exploration and prevent getting the system stuck in local minima, random walk actions are selected with probability P_{rw} .

As shown in Fig. 8 (a)–(c) the value of η produces significant variation in the expected response time of the system across the offered load range (at 3, 5 and 7 req/s). A larger η reduces the average response time because it allows the CNC to quickly learn the actual channels' state. Giving low importance to the most recent samples (low α) tends to produce better results. However, the performance differences for different values of α were not too high, except for the highest traffic load. No major performance differences were observed

with different values of α for the low and mid traffic cases, unlike the high traffic case. Throughput performance was almost indistinguishable with different values of α or η as shown in Fig. 8 (d).

Fig. 9 depicts the impact of the exploration vs. exploitation tradeoff, in terms of the probability P_{rw} of selecting a random channel rather than the SNN decision for a file transfer. Fig. 9 (a)–(c) show the response time for $\lambda = 3, 5$, or 7 . Fig. 9 (d) shows the corresponding throughput, which we observed to be invariant across different choices of α . For the selected scenario, a larger exploration (higher value of P_{rw}) produced detrimental performance except for small values of P_{rw} (in the 0–0.05 range) where we observed a small improvement with better exploration for the mid-to-high traffic intensity. In general, we observed that a value in the range 0.05–0.2 produced a reasonable balance and the best results for this scenario.

VI. Related Works

The present work is possibly one of the earliest efforts, if not the first, to relate the time-to-fire of a spiking neural network to a cognitive networking problem. However, other techniques have been explored for related purposes in the literature. Reinforcement learning has been a popular approach [6], [7] that provides some degree of artificial cognition. It has been recently applied to the routing and scheduling problem in challenged networks [8]. Related network routing problems have been addressed with supervised learning and artificial neural networks of continuous activation neurons. The recurrent structure of a Hopfield neural network and variations of this approach have provided the associative memory support required to store routing decisions [9]–[14]. Several works have demonstrated the possibility of using both shallow feed-forward neural networks for routing [15]–[19] and deep learning within the context of software defined routing with traffic prediction [20]. While suitable for their intended application, a main drawback of most of these methods is in the central computation assumption, which limits their suitability for autonomous networks. Another related work is the Cognitive Packet Network (CPN) [21], [22], which uses a distributed approach to packet routing. CPN uses reinforcement learning to adjust weights of Random Neural Networks (RNN) which define the routing control of the approach.

As neuromorphic low-power systems gain momentum, the possibility of information encoding in the time-to-fire of spikes of SNN (spiking neural networks) reduces the computational effort involved in the use of these models, bringing processing efficiency and speed to their applications. Several learning methods for SNN have been proposed in recent years. Spike timing-dependent synaptic plasticity (STDP) links patterns of presynaptic and postsynaptic activity to changes in the synaptic strength [23] as an asymmetric form of Hebbian learning. It has been applied to control a virtual insect [24]. Hebbian learning has been also applied to SNN [25] using a similar Leaky-Integrate-and-Fire model that we use in this work. Variations of the training method include the use of genetic algorithms (GA) in a mobile robot application [26].

VII. Conclusion

We have presented a spiking neural network to provide learning and adaptation features for cognitive networks. The proposed Cognitive Network Controller (CNC) exploits the spatial-temporal dynamics of SNNs to transform perceived network state to network control actions. The CNC encodes actions in the time-to-fire of spikes. To find a learning rule for the CNC, considering the electrical properties of spiking neurons, neural network structure, and external stimuli, we defined an algorithm that determines the effective range of synaptic strengths. By adapting SNN's synapse strength values in relation to performance outcomes of prior actions, the CNC continuously learns how to optimize future action decisions to achieve a desirable objective.

The evaluation of a simulated multi-link space communication of time-varying performance demonstrated the effectiveness of our approach. In that scenario, the CNC selects a channel for each bundle transmission attempting to achieve the service level objective (SLO) of average response times to be less than a second. The results indicate that the CNC achieves lower response times than common methods without impacting throughput. We observed up to 10 times lower delays than the reflex (greedy) method and many more times lower than the random and round robin methods.

This work is one of the first in the field to explore the use of SNNs for cognitive networking. Further work is needed, but we demonstrated that the approach is feasible. We revealed deficiencies of the CNC to achieve low delays under very high traffic. In those cases, packet congestion caused extreme delays in the feedback flow for the CNC, which impacted its learning ability and produced detrimental channel selection performance. Future research will address this limitation, possibly by exploring the adoption of application-specific mechanisms that are less sensitive to congestion and by switching the resource selection method after exceeding a certain load level. Future work will further extend the scope of the CNC and compare its performance with more elaborate methods.

ACKNOWLEDGMENT

The author would like to express his thanks to Gilbert Clark for providing suggestions on this research.

This work was supported by an Early Career Faculty grant from NASA's Space Technology Research Grants Program. The associate editor coordinating the review of this paper and approving it for publication was F. Ye.

Biography

Ricardo Lent (M'92–SM'14) received the B.Sc. degree in electronic engineering from Ricardo Palma University, Peru, the M.Sc. degree in telecommunications engineering from the Universidad Nacional de Ingeniería, Peru, and the M.Sc. and Ph.D. degrees in computer

science from the University of Central Florida, USA, in 2003. He is currently an Assistant Professor with the Department of Engineering Technology, University of Houston. His prior experience includes work as a Research Fellow with the Department of Electrical and Electronic Engineering, Imperial College London, U.K., a Principal Network Engineer in the industry, and as a Visiting Assistant Professor with the University of Central Florida. His current research interests include cognitive networking, mobile and pervasive communications, and green computing.

REFERENCES

- [1]. Maas W, "Networks of spiking neurons: The third generation of neural network models," *Trans. Soc. Comput. Simulat. Int.*, vol. 14, no. 4, pp. 1659–1671, 12 1997 [Online]. Available: <http://dl.acm.org/citation.cfm?id=281543.281637>
- [2]. Paugam-Moisy H and Bohte SM, "Computing with spiking neuron networks," in *Handbook of Natural Computing*. Heidelberg, Germany: Springer-Verlag, 2012, pp. 335–376.
- [3]. Volanis G, Antonopoulos A, Hatzopoulos AA, and Makris Y, "Toward silicon-based cognitive neuromorphic ICs—A survey," *IEEE Des. Test*, vol. 33, no. 3, pp. 91–102, 6 2016.
- [4]. Gerstner W and Kistler W, *Spiking Neuron Models: An Introduction*. New York, NY, USA: Cambridge Univ. Press, 2002.
- [5]. The Interplanetary Overlay Network (ION) Software Distribution. ION-DTN. Accessed: Apr. 1, 2018. [Online]. Available: <https://sourceforge.net/projects/ion-dtn>
- [6]. Arroyo-Valles R, Alaiz-Rodriguez R, Guerrero-Curieses A, and Cid-Sueiro J, "Q-probabilistic routing in wireless sensor networks," in *Proc. 3rd Int. Conf. Intell. Sensors Sensor Netw. Inf.*, 12 2007, pp. 1–6.
- [7]. Littman M and Boyan J, "A distributed reinforcement learning scheme for network routing," *School Comput. Sci*, Carnegie Mellon Univ, Pittsburgh, PA, USA, Rep. CMU-CS-93-165, 1 1993.
- [8]. Dudukovich R, Hylton A, and Papachristou C, "A machine learning concept for DTN routing," in *Proc. IEEE Int. Conf. Wireless Space Extreme Environ. (WiSEE)*, 10 2017, pp. 110–115.
- [9]. Dixon MW, "An optimal neural routing algorithm," in *Proc. Singapore Int. Conf. Inf. Eng. Commun. Netw.*, vol. 2 Singapore, 9 1993, pp. 852–856.
- [10]. Liu W and Wang L, "Solving the shortest path routing problem using noisy Hopfield neural networks," in *Proc. WRI Int. Conf. Commun. Mobile Comput.*, vol. 2, 1 2009, pp. 299–302.
- [11]. Liu B, Gao J, and Xie X, "Optimized routing design for wireless mesh networks based on neural network," in *Proc. 3rd Int. Conf. Anti Counterfeiting Security Identification Commun.*, 8 2009, pp. 552–555.
- [12]. Bastos-Filho CJA, Schuler WH, Oliveira ALI, and Vitorino LN, "Routing algorithm based on swarm intelligence and Hopfield neural network applied to communication networks," *Electron. Lett.*, vol. 44, no. 16, pp. 995–997, 7 2008.
- [13]. Ouyang YC and Bhatti AA, "Neural network based routing in computer communication networks," in *Proc. IEEE Int. Conf. Syst. Eng.*, 8 1990, pp. 621–624.
- [14]. Castellano G, Attolico G, Stella E, and Distanto A, "Routing applications of the Hopfield neural network," in *Proc. 8th Mediterr. Electrotech. Conf. Ind. Appl. Power Syst. Comput. Sci. Telecommun. (MELECON)*, vol. 3, 5 1996, pp. 1457–1460.
- [15]. Mekinda L and Muscariello L, "Supervised machine learning-based routing for named data networking," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 12 2016, pp. 1–6.
- [16]. Mazare A, Ionescu L-M, Lita A-I, Serban G, and Ionut M, "Mobile system with real time route learning using hardware artificial neural network," in *Proc. 7th Int. Conf. Electron. Comput. Artif. Intell. (ECAI)*, 6 2015, pp. 45–48.
- [17]. Tur anik M, "Network routing by artificial neural network," in *Proc. Military Commun. Inf. Syst. Conf. (MCC)*, 10 2012, pp. 1–5.

- [18]. Zahid M, Soperi M, Haider M, and Bakar KA, "Scheduling of OSPF routing table calculation using generalized regression neural network," in Proc. 17th IEEE Int. Conf. Netw, 12 2011, pp. 311–315.
- [19]. Baldo N, Tamma BR, Manoj BS, Rao RR, and Zorzi M, "A neural network based cognitive controller for dynamic channel selection," in Proc. Int. Conf. Commun, 6 2009, pp. 1–5.
- [20]. Azzouni A, Boutaba R, and Pujolle G, "NeuRoute: Predictive dynamic routing for software-defined networks," in Proc. 13th Int. Conf. Netw. Service Manag. (CNSM), 11 2017, pp. 1–6.
- [21]. Gelenbe E, Lent R, Montuori A, and Xu Z, "Cognitive packet networks: QoS and performance," in Proc. 10th IEEE Int. Symp. Model. Anal. Simulat. Comput. Telecommun. Syst. (MASCOT), 2002, pp. 3–9.
- [22]. Lent R, "Linear QoS goals of additive and concave metrics in ad hoc cognitive packet routing," IEEE Trans. Syst., Man, Cybern., Syst, vol. 36, no. 6, pp. 1255–1260, 12 2006.
- [23]. Farries MA and Fairhall AL, "Reinforcement learning with modulated spike timing dependent synaptic plasticity," J. Neurophysiol, vol. 98, no. 6, pp. 3648–3665, 2007. [PubMed: 17928565]
- [24]. Mazumder P et al., "Digital implementation of a virtual insect trained by spike-timing dependent plasticity," Integr. VLSI J, vol. 54, pp. 109–117, 6 2016.
- [25]. Gupta A and Long LN, "Hebbian learning with winner take all for spiking neural networks," in Proc. Int. Joint Conf. Neural Netw, 2009, pp. 1189–1195.
- [26]. Alnajjar F and Murase K, "Self-organization of spiking neural network generating autonomous behavior in a real mobile robot," in Proc. Int. Conf. Comput. Intell. Model. Control Autom, vol. 1, 2005, pp. 1134–1139.

Fig. 1.

Leaky-Integrate-and-Fire neuron model. The presynaptic spike $\delta(t - t_j^{(f)})$ emitted by neuron j arrives after the low-pass filter at the postsynaptic neuron i . If the voltage across the RC circuit reaches θ , neuron i emits its f -th pulse $\delta(t - t_i^{(f)})$.

Fig. 2.

SNN implementing an 8-action CNC. Solid lines: excitatory synapses; bold lines: synapses in both directions; dashed lines: inhibitory synapses. I_{ref} , $E1$ – $E8$, and $I1$ – $I8$ indicate the external stimuli.

Fig. 3.

(a) Postsynaptic activity (bottom) caused by 4 presynapses of strength 30 (top 4 lines) and excited with 2.1 nA from time 0 s; (b) effect of a single-terminal synapse weight on the postsynapse time-to-fire with threshold of -50mV , LIF unit resistance: $10\text{M}\Omega$ and τ : 10mS . Dashed lines: 0V .

Fig. 4.

The four parallel links between the ground station (Houston) and the ISS are only available for about 5 minutes during the pass. The distance between the stations varies rapidly by more than 400 km during that time interval, causing great variability in communication performance.

Fig. 5.

Response time of file transfers with LTP segment sizes: (a) 1000 bytes, (b) 2000 bytes, with a SLO of 1-second response time limit. Results for the CNC/SNN were obtained with $P_{rw} = 0.1$, $\eta = 1000$, and $\alpha = 0.1$.

Fig. 6.
Throughput vs. offered load with LTP segment sizes of (a) 1000 bytes or (b) 2000 bytes.

Fig. 7.

Response time (a)–(c) and throughput (d) with 1000-byte LTP segment lengths. Results for the CNC/SNN were obtained with $P_{rw} = 0.01$, $\eta = 1000$, and $\alpha = 0.1$.

Fig. 8.

Response time results for workload levels of 3, 5, and 7 req/s vs. learning factor η . We observe small sensitivity to α (short term memory), but larger values of η tend to deliver the lowest delays. Throughput results (d) show that different values of α and η produce no major impact.

Fig. 9.

Response time results for 3, 5, and 7 req/s vs. random walk probability show large sensitivity to P_{rw} and small sensitivity to α across load levels. Throughput was almost constant except for the highest load.

TABLE I**SUMMARY OF SNN MODEL PARAMETERS**

Parameter	Value
SNN simulation time	60 mS
SNN simulation step	0.5 mS
Time constant (τ)	10 mS
Circuit model Resistance (R)	10 M Ω
Maximum depolarization (spike voltage)	35 mV
Spike fire threshold (θ)	-50 mV
Reset voltage (after firing a spike)	-90 mV
Rest potential	-70 mV
Synapse strength range (obtained by auto-calibration)	[79.76, 206.89]
Random walk probability (P_{rw})	0.01–0.5
Learning factor (η)	1–2000
Exponential average parameter (α)	0.1,0.5,0.9