Journal of Network and Computer Applications 106 (2018) 68–77

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

A hardware testbed for learning-based spectrum handoff in cognitive radio networks

Koushik A.M.[a], Elizabeth Bentley[b], Fei Hu[a,*], Sunil Kumar[c]

[a] Electrical and Computer Engineering, The University of Alabama, USA
[b] Air Force Research Laboratory, Rome, New York, USA
[c] Electrical and Computer Engineering, San Diego State University, San Diego, CA, USA

ARTICLE INFO

Keywords:
Spectrum handoff
Transfer learning
Reinforcement learning
Q learning
Cognitive Radio Network (CRN)
GNU radio
USRP
Testbed

ABSTRACT

A real-time cognitive radio network (CRN) testbed is implemented by using the universal software radio peripheral (USRP) and GNU Radio to demonstrate the use of reinforcement learning and transfer learning schemes for spectrum handoff decisions. By considering the channel status (idle or occupied) and channel condition (in terms of packet error rate), the sender node performs the learning-based spectrum handoff. In reinforcement learning, the number of network observations required to achieve the optimal decisions is often prohibitively high, due to the complex CRN environment. When a node experiences new channel conditions, the learning process is restarted from scratch even when the similar channel condition has been experienced before. To alleviate this issue, a transfer learning based spectrum handoff scheme is implemented, which enables a node to learn from its neighboring node(s) to improve its performance. In transfer learning, the node searches for an expert node in the network. If an expert node is found, the node requests the Q-table from the expert node for making its spectrum handoff decisions. If an expert node cannot be found, the node learns the spectrum handoff strategy on its own by using the reinforcement learning. Our experimental results demonstrate that the machine learning based spectrum handoff performs better in the long term and effectively utilizes the available spectrum. In addition, the transfer learning requires less number of packet transmissions to achieve an optimal solution, compared to the reinforcement learning.[1]

1. Introduction

The cognitive radio network (CRN) is considered as a promising solution to address the issue of spectrum scarcity and effective spectrum utilization. In CRN, the secondary users (SU) are allowed to occupy the spectrum when it is not used by the primary users (PU), which is known as the dynamic spectrum access (DSA) (Crohas, 2008). However, the frequent interruptions from PUs in CRN force the SUs to perform handoff to other idle channels. The spectrum handoff can also occur due to node mobility (Wang and Wang, 2008; Song and Xie, 2011; Kumar et al., 2016). Thus, it is very important for SUs to keep monitoring the link status (due to temporal mobility) and link quality (spatial mobility).

In this paper, we implement the spectrum handoff process in a CRN testbed using the universal software radio peripheral (USRP) boards and GNU Radio. Our main goal is to enable each SU node to learn the spectrum handoff based on its past observations. Since CRN is able to learn and reason the radio environment through a cognitive engine, the use of machine learning algorithms can enhance the learning and reasoning of the spectrum handoff process. Here, a learning model represents the process of acquiring the knowledge by interacting with the environment, to improve the future decisions. In recent years, the machine learning algorithms have been widely used in CRN (Hossain et al., 2014; Bkassiny et al., 2013).

In this paper, we implement the reinforcement learning (RL) and transfer learning (TL) based spectrum handoff schemes in CRN, by using the GNU Radio programming environment (Ke-Yu and Chen, 2006) for multimedia transmissions (such as real-time video). Note that a myopic spectrum handoff scheme may not achieve the best performance in the long-term, since it tends to select the channels which maximize the short-term reward. When an SU learns about the spectrum handoff decisions on its own by using the RL algorithm (Wu et al., 2014), it typically needs more time to converge to the optimal solution, which is undesirable for real-time data transmission. Instead, a new node can seek help from other nodes in the network, which are termed as the 'expert nodes' (Galindo-Serrano et al., 2010; Giupponi et al., 2010; Dohler et al., 2010). Specifically, when a new (or learning) node joins the network, it searches for an expert node in the network by

* Corresponding author.
[1] DISTRIBUTION STATEMENT A: Approved for Public Release; distribution unlimited 88ABW-2017–6274; 13 Dec 2017.

https://doi.org/10.1016/j.jnca.2017.11.003
Received 16 August 2017; Received in revised form 2 October 2017; Accepted 10 November 2017
Available online 31 January 2018
1084-8045/ © 2018 Elsevier Ltd. All rights reserved.

using the control channel. If an expert node is found, it shares its optimal strategy with the new node to help with the spectrum handoff decisions. This is termed as the 'transfer learning' (TL). When the communication tasks are similar between the learning and expert nodes, the knowledge transferred from the expert node enables the learning node to start communications from the optimal condition without taking much time to acquire knowledge about the RF environment, which significantly enhances its performance. If there is no expert node in the network, the new node learns about the environment on its own and builds the optimal strategy by using RL.

We address the following issues in building a hardware testbed for intelligent spectrum handoff. (*i*) *How often should the node sense the channel?* (*ii*) *How often should the learning algorithm be updated?* (*iii*) *How long should the learning node wait for the response from the expert node?* Since the GNU Radio software does not have any predefined machine learning functions, all the modules need to be built from the scratch. The main contributions of this paper are twofold:

(1) *Real-time CRN Testbed*: The USRP and GNU Radio based testbed, which uses the directional antennas, is built for multimedia data transmissions. Using USRP 210 series we have implemented spectrum sensing, spectrum handoff and other CRN functions. All the communication modules are built using Python and C ++ in the GNU Radio environment. In addition, we have implemented the machine learning modules using Python on the host level of GNU Radio. Thus our CRN testbed serves as a platform for implementing the advanced CRN protocols and machine learning algorithms.
(2) *TL-based spectrum handoff*: RL is used when the node is new to the network and cannot find an expert node. In our testbed, we use the Q-learning as the RL scheme to perform spectrum handoff due to its ability to explore and exploit the best actions for each state. To enhance the process of adaptation to the radio environment and to achieve optimal condition faster, we have implemented TL algorithm. There are several TL approaches, such as the inverse RL, apprenticeship learning, etc. We have used a typical Docitive learning model, where the optimal Q-table is transferred from the expert node to the learning node.

The rest of this paper is organized as follows: The related work is summarized in Section II. The RL and TL based spectrum handoff schemes are explained in Sections III and IV, respectively. The CRN testbed set up and design challenges are described in Section V. The experimental results are presented in Section VI, followed by the conclusions in Section VII.

2. Related work

Several CRN testbeds using USRP and GNU Radio, with spectrum sensing, dynamic spectrum access and interference management functions, have been discussed in (Crohas, 2008; Patcha, 2011; Aftab and Mufti, 2011). A CRN testbed with spectrum sensing function was implemented in (Ke-Yu and Chen, 2006). The authors also extended their work to observe the burst errors in OFDM using Markov traffic models, and implemented a 4-node CRN network to observe the effect of interference on the delay performance. A comparative study of different spectrum sensing techniques using the USRP and GNU Radio was performed in (Galindo-Serrano et al., 2010). Researchers at UC Berkeley (Mishra et al., 2005) designed a CRN testbed by using BEE2 and a multi-FPGA emulation engine, to verify different sensing processes at the physical layer in real-time system. They developed two CRN testbeds with 8 WARP nodes and 11 USRP nodes. A large-scale CRN testbed with distributed spectrum sensing was developed in (Qiu et al., 2012). The researchers at Virginia Tech (Newman et al., 2009) developed the VT-CORNET testbed, for the development, testing and evaluation of several cognitive radio applications.

However, only a few testbeds have used the machine learning algorithms in CRN. Authors in (Anil et al., 2014) developed machine learning plugins (i.e., linear logistic regression classifiers), by using the GNU Radio companion and Python coding. A Q-learning based interference management in cognitive femtocell networks was developed in (Elsayed and Mohamed, 2015) using the USRP and GNU Radio. A practical signal detection and classification scheme in GNU Radio was developed in (O'shea et al., 2007) by using the artificial neural networks (ANN). The fusion of signal detection and classification algorithms. A Q-learning based channel allocation platform was proposed in (Hosey et al., 2009) for a 4-node CRN, where each node acts individually without collaborating with other nodes, to avoid the overhead introduced in cooperative spectrum access. In (Ren et al., 2010), a Q-learning based spectrum management system for a Markovian modeled, large-scale multi-agent network was implemented, and the success rate of packet transmission was improved.

Most of the existing testbeds have used the RL based models. Ours is the first testbed to implement a transfer learning algorithm to enhance the learning speed of the network, which can tune its strategy to the dynamic variations of the channel.

3. Reinforcement learning for spectrum handoff

The RL (Sutton and Barto, 1998) is a prominent unsupervised learning schemes, which can enable a node to learn autonomously in CRN environment (Bkassiny et al., 2013; Sunderland, 2010). The RL is a special case of the Markov decision process (MDP), which can be stated as a tuple *(S, A, T, R)*. Here *S* corresponds to the finite set of states for the node; *A* is the finite set of actions available for a node in each state; *T* defines the transition probability, $(s_j|s_i, a)$, from state s_i to state s_j, as a result of action $a \in A$; R denotes the reward, $R(s, a)$, observed when action, $a \in A$ is performed while in state, $s \in S$. After a series of actions, $a \in A$ for state, $s \in S$, the system reaches an optimal condition by building an optimal policy $\pi(s,a)$, which defines the probability of taking an action a in state s.

In our testbed, the tuple (S, A, T, R) is defined as follows:

<u>*States, S*</u>: When occupying a channel at iteration *t*, the node observes the state as $\{\xi_t, \phi_t\}$. Here, ξ_t denotes the channel condition (in terms PER) and ϕ_t denotes the channel status (idle or busy).

<u>*Action, A:*</u> The actions considered for spectrum handoff are: (1) Stay and transmit in the same channel; (2) Perform spectrum handoff to another vacant channel upon interruption from PU, or when channel condition deteriorates.

<u>*Reward, R*</u> is defined as the immediate reward incurred for the multimedia transmission. When the channel condition is good and the action taken is transmission (a_1), we assign the reward of 10. When channel condition is bad and action taken is spectrum handoff (a_2), the reward is 5. For any other combination of state and action the reward is −5, which indicates that the action was not desired in the observed state.

<u>*Transition probability, T*</u> cannot be determined since the radio environment is dynamic.

We have adopted the Q-learning based reinforcement learning. The Q-learning algorithm estimates the Q-values, $Q(s, a)$, for the joint state-action pairs (s, a). The Q-table determines how good it is for a given agent to perform a certain action under a given state. The one-step Q-table update equation is defined as,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma maxQs_{t+1}, a - Q(s_t, a_t] \quad (1)$$

where s_t, a_t, andr_t are the state, action, and reward, respectively, at the t^{th} iteration of the learning process. Here, $\gamma \in (0,1)$ is the discount factor that maximizes the total expected rewards by controlling the influence of previous reward on current action; $0 \le \alpha \le 1$ is the learning rate defining how much of the newly acquired information can be used for strengthening the Q-value so that it attains the optimal value, Q^*. All the state-action pairs need to be updated continuously to achieve the