SMART: A Spectrum-Aware Cluster-based Routing Scheme for Distributed Cognitive Radio Networks

Yasir Saleem[a,b,*], Kok-Lim Alvin Yau[a], Hafizal Mohamad[b], Nordin Ramli[b], Mubashir Husain Rehmani[c]

[a] *Faculty of Science and Technology, Sunway University, Selangor, Malaysia*
[b] *Wireless Network and Protocol Research Lab, MIMOS Berhad, Kuala Lumpur, Malaysia*
[c] *COMSATS Institute of Information Technology, Wah Cantt, Pakistan*

Abstract

Cognitive radio (CR) is the next-generation wireless communication system that allows unlicensed users (or secondary users, SUs) to exploit the under-utilized spectrum (or white spaces) in licensed spectrum whilst minimizing interference to licensed users (or primary users, PUs). This article proposes a SpectruM-Aware clusteR-based rouTing (SMART) scheme that enables SUs to form clusters in a cognitive radio network (CRN) and enables each SU source node to search for a route to its destination node on the clustered network. An intrinsic characteristic of CRNs is the dynamicity of operating environment in which network conditions (i.e., PUs' activities) change as time goes by. Based on the network conditions, SMART enables SUs to adjust their cluster size, which represents the number of nodes in a cluster, and searches for a route on the clustered network using an artificial intelligence approach called reinforcement learning. Simulation results show that SMART selects stable routes and significantly reduces interference to PUs, as well as, routing overhead in terms of route discovery frequency without significant degradation of throughput and end-to-end delay.

Keywords: Cognitive radio, clustering, routing, reinforcement learning, cluster merging, cluster splitting

*Corresponding author
Email address: 13032057@imail.sunway.edu.my (Yasir Saleem)

1. Introduction

Cognitive radio (CR) [1] is the next-generation wireless communication system that enables unlicensed or secondary users (SUs) to explore and use underutilized licensed spectrum (or white spaces) owned by licensed or primary users (PUs) in order to improve the overall spectrum utilization. PU has licensed channel for certain time and frequency during which it can use its channel without interference from other users in the network. SU uses under utilized channels of PUs. During the communication of SU, if PU reappears on its channel, SU has to vacate this channel and switch to another available channel.

Routing is a fundamental function of any wireless network which enables data communication by finding a route from source node to destination node across the network. Routing in CRN is challenging due to several reasons. For instance, firstly, CRN is characterized by the dynamicity of channel availability due to different levels of PUs' activities, which varies the amount of white spaces. Secondly, multiple channels exist in CRNs which are heterogeneous in nature, therefore, it is challenging for SUs to select the most appropriate channels from a list of available channels. Thirdly, the dynamicity of channel availability causes lack of common control channel for control information exchange in routing. Fourthly, the availability of multiple heterogeneous channels and dynamicity of channel availability may cause frequent channel switching by SU, which can degrade SUs' network performance. Therefore, routing protocols for traditional wireless networks that maintain end-to-end paths, such as ad-hoc on-demand distance vector (AODV) routing protocol, are not preferable for CRNs because they do not consider the challenges of routing in CRNs and may cause high network overhead by constant flooding of routing messages. Hence, such routing protocols cannot be directly applied in CRNs. Routing protocols for CRNs must address the challenges of CRNs and be spectrum-aware, so that routes should be stable and SUs can perform data communication for longer period of time without much disruptions, as well as minimize interference to PUs.

Routing protocols can be cluster-based which runs over the clustered network. Clustering is a topology management mechanism which organizes nodes into logical groups called clusters. Cluster-based routing is preferred in CRN for the following reasons. Firstly, it provides network scalability by reducing the flooding of routing control messages, such as route request (RREQ) and route reply (RREP), throughout the network, since routing

control messages are only exchanged among some nodes such as clusterheads and connecting nodes among clusters (or gateway nodes). Secondly, it provides network stability by reducing the effects of dynamic conditions in the network (i.e., PUs' activities), since the changes affect the network at cluster level, so only local update is required instead of whole network reconfiguration. Thirdly, it supports cooperative tasks and improves channel sensing outcomes. For example, a clusterhead collects channel sensing outcomes from its member nodes and subsequently makes final decision on channel availability. This improves the accuracy of channel availability decision as compared to the decision made on the outcome of single node.

Reinforcement learning (RL) [2] is an artificial intelligence approach that enables a decision maker to observe its state and reward, learn, and subsequently carry out a proper action so that the state and reward, which are the consequences of the action, improve in the next time instance. RL has been applied in wireless networks which enables each SU to observe, learn, and make the right decisions on routing (i.e., route selection) in order to maximize network performance.

This article presents SpectruM-Aware clusteR-based rouTing (SMART), which is a cluster-based routing scheme, that selects stable routes and maximizes SUs' routing and clustering performances including SUs' interference to PUs and cluster size without significant degradation of throughput and end-to-end delay. SMART applies an artificial intelligence approach called reinforcement learning (RL) that enables each SU to observe, learn, and make the right decisions on routing (i.e., route selection) in order to maximize network performance.

SMART provides three main contributions imperative to CRNs as follows:

C.1 SMART maximizes the utilization of white spaces in order to maximize SUs' network performance. This can be achieved through adaptation to the dynamicity of network conditions (i.e., PUs' activities).

C.2 SMART aims to fulfill requirement on the minimum number of common channels in a cluster, which enhances cluster stability through improved connectivity among nodes in a cluster, in order to increase the availability of at least a single common channel in a cluster, while leaving the rest of the common channels as backups. A common channel is essential for member nodes to send data packets to their respective clusterheads efficiently so that clusterheads do not switch channels constantly. A group of geographically adjacent SUs tend to share a similar

set of available channels, so smaller cluster size increases the number of common channels in a cluster [3]. Higher number of common channels in a cluster prevents frequent cluster splitting as a result of the re-appearance of PUs' activities.

C.3 SMART solves one of the main problems of broadcasting using single transceiver in CRNs, which requires a SU to send similar packets in various channels so that all neighboring SUs can receive the packet. In SMART, using a single transceiver, a clusterhead knows about its neighboring clusters through gateway nodes. Firstly, it broadcasts routing control messages (i.e., RREQ and RREP) using its operating channel. Secondly, gateway nodes that receive the routing control messages forward them to their respective neighboring clusters using the operating channel of the neighboring clusters. Thus, the clusterhead and gateway nodes are aware of the operating channels of neighboring nodes, which allow them to broadcast efficiently without broadcasting on all the available channels.

The rest of this paper is organized as follows. Section 2 presents background on clustering and RL. Section 3 presents related work. Section 4 presents system model. Section 5 presents SMART clustering scheme (i.e., channel capacity metric, cluster formation, gateway node selection, cluster merging and cluster splitting). Section 6 presents SMART routing scheme. Section 7 presents performance evaluation, results and discussion. Section 8 presents conclusion and future work.

2. Background

This section presents background on clustering in CRNs and RL.

2.1. Clustering in Cognitive Radio Networks

Clustering, which is a topology management mechanism, has traditionally been applied in wireless networks to organize nodes into logical groups in order to provide network scalability and stability (see Section 1). Popular traditional clustering algorithms, such as lowest ID [4] and maximum node degree [5], may not be suitable in CRNs due to the dynamicity of channel availability and the presence of multiple channels. Among nodes in a single-hop or multi-hop neighborhood, the lowest ID clustering algorithm selects a node with the lowest ID as the clusterhead; while the maximum node degree

clustering algorithm selects a node with the highest number of neighbor nodes as the clusterhead.

The cluster structure is comprised of clusterheads and member nodes, and it provides a suitable network model to support cooperative tasks, such as channel sensing and routing, which are essential to CR operations. As an example, a clusterhead collects channel sensing outcomes from its member nodes and subsequently makes final decisions on channel availability. This has been shown to improve the accuracy of channel sensing outcomes in the presence of channel fading and shadowing which may cause the detection of PUs' activities to failure [6].

Figure 1 shows an example of a cluster structure in which nodes in a CRN are grouped. There are three clusters (i.e., C_1, C_2 and C_3). Each cluster is comprised of four kinds of nodes, namely clusterhead, member node, relay node and gateway node. A clusterhead (i.e., CH_1, CH_2 and CH_3) serves as a local point of process for various applications such as channel sensing and routing. A member node (i.e., $MN_{1,1}$, $MN_{2,1}$ and $MN_{3,1}$) associates itself with a clusterhead. For instance, member nodes $MN_{1,1}$, $MN_{2,1}$ and $MN_{3,1}$ are associated with clusterhead CH_1 in cluster C_1. Clusterhead and member nodes communicate regularly among themselves using a common channel, and these are called intra-cluster communications. The common channel is available to all member nodes of a cluster. A relay node (i.e., $RN_{1,2}$) is a member node that provides connectivity to a member node which is located out of the range of clusterhead. For instance, relay node $RN_{1,2}$ provides connectivity to member node $MN_{1,2}$ towards clusterhead CH_2 in cluster C_2. A gateway node, which is also a member node located at the fringe of a cluster, can hear from neighboring cluster(s), and so it provides inter-cluster communications. As an example, gateway node $GN_{1,3,2}$ is associated with clusterhead CH_3, and it provides two-hop inter-cluster connectivity from CH_3 to neighboring clusterhead CH_2. As another example, gateway node $GN_{1,1,2}$ is associated with clusterhead CH_1, and it provides three-hop inter-cluster connectivity from CH_1 to neighboring clusterhead CH_2. The clusterheads and gateway nodes form a backbone to the SU base station (SU BS). For instance, in Figure 1, member nodes $MN_{2,1}$ and $MN_{3,1}$ send data packets to destination SU BS through backbone CH_1-$GN_{1,1,2}$-$GN_{1,2,1}$-CH_2-$GN_{2,2,3}$-CH_3-$GN_{2,3,\mathrm{BS}}$. The number of hops between a member node and a clusterhead in a cluster may be a single [7], two or more.

Cluster size, which represents the number of nodes in a cluster, affects various performance metrics. Larger cluster size reduces routing overhead

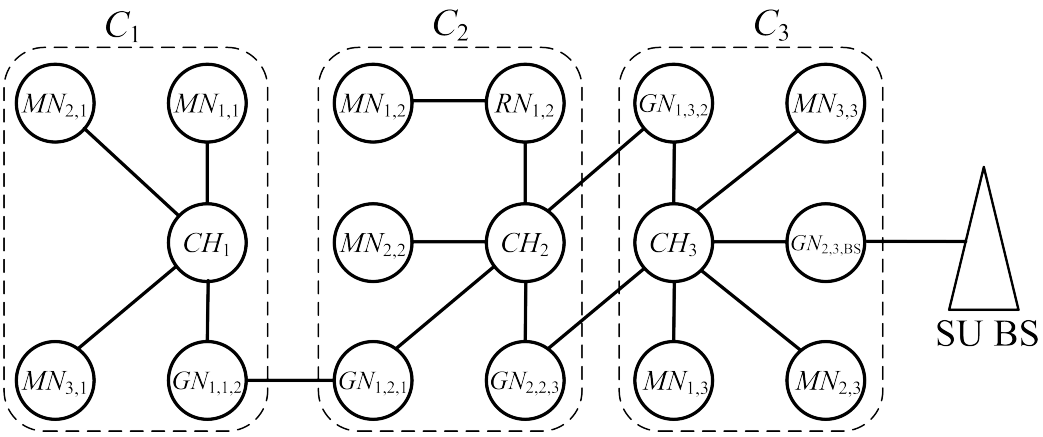

Figure 1: A cluster structure in a CRN.

since the flooding of routing overheads only involves clusterheads and gateway nodes along a backbone, as well as reduces error probability in the final decision on channel availability, since this decision is made based on channel sensing outcomes collected from higher number of nodes in a cluster. Smaller cluster size (or larger number of clusters in a network) increases the number of common channels, and hence connectivity among nodes in a cluster, because physically close nodes may share a similar set of available channels. Since clusters may use different common channels, the contention and interference levels in the network can be reduced, and this subsequently improves routing performance. Higher number of common channels in a cluster minimizes the occurrence of re-clustering due to improved connectivity among nodes in a cluster. While achieving larger cluster size may seem to be more favourable in traditional distributed networks in order to improve scalability, the same cannot be said for CRNs since achieving smaller cluster size improves stability and addresses the intrinsic characteristics of CRNs, particularly the dynamicity of channel availability. SMART adjusts cluster size based on network performance brought about by routing, which is dependent on network conditions (i.e., PUs' activities) that change as time goes by, so that a cluster fulfils the requirement on cluster size to improve scalability and the number of common channels in a cluster to improve stability.

SMART provides two-fold functions: cluster maintenance and routing. Cluster maintenance adjusts cluster size through cluster merging and cluster splitting mechanisms; while routing searches for a route from a SU source

node to a SU destination node on the clustered network using reinforcement learning. Both cluster maintenance and routing mechanisms can best be explained using an example. In Figure 1, suppose, member nodes $MN_{2,1}$ and $MN_{3,1}$ send data packets to destination SU BS through backbone CH_1-$GN_{1,1,2}$-$GN_{1,2,1}$-CH_2-$GN_{2,2,3}$-CH_3-$GN_{2,3,\text{BS}}$. Due to the dynamicity of network conditions (i.e., PUs' activities), clusterhead CH_1's network performance (i.e., packet loss rate) deteriorates, and so it must be adaptive to the changing environment (see C.1 in Section 1). Suppose, there is lack of a common channel in cluster C_1 which causes lack of connectivity among nodes in the cluster, and so a large number of packets expire and are dropped at member node $MN_{2,1}$. Cluster C_1 undergoes cluster splitting and a new cluster C_4 is formed as shown in Figure 2. Subsequently, clusterhead CH_1 may use existing route or search for a new route while the newly-formed clusterhead CH_4 must search for a new route. In Figure 2, upon completion of the routing phase, clusterhead CH_1 uses existing route CH_1-$GN_{1,1,2}$-$GN_{1,2,1}$-CH_2-$GN_{2,2,3f}$-CH_3-$GN_{2,3,\text{BS}}$ to its destination SU BS; and clusterhead CH_4 uses a new route CH_4-$GN_{1,4,2}$-$GN_{3,2,4}$-CH_2-$GN_{2,2,3}$-CH_3-$GN_{2,3,\text{BS}}$. Note that, clusters C_1 and C_4 may use different common channels and so the contention and interference levels in the network can be reduced. This also means that, since the contention and interference levels are lower among clusters C_1 and C_4 in Figure 2 compared to a single cluster C_1 in Figure 1, network performance is expected to improve. Forming smaller clusters is favorable because it increases the number of common channels in a cluster and so it improves stability (see C.2 in Section 1); however, smaller clusters may not be favorable due to higher error probability in sensing outcomes of channel sensing, and so the requirement on the minimum number of nodes in a cluster must be fulfilled to improve scalability.

2.2. Reinforcement Learning

Reinforcement learning (RL) [2] is an artificial intelligence approach that enables a decision maker (or agent) to observe its state and reward, learn, and subsequently carry out a proper action so that the state and reward, which are the consequences of the action, improve in the next time instance.

Q-learning [2] is a popular technique in RL. The important representations in the RL model for an agent are state, action and reward. Denote decision epochs by $t \in T = 1, 2, ...$, the knowledge possessed by agent n for a particular state-action pair at time t is represented by Q-function as follows:

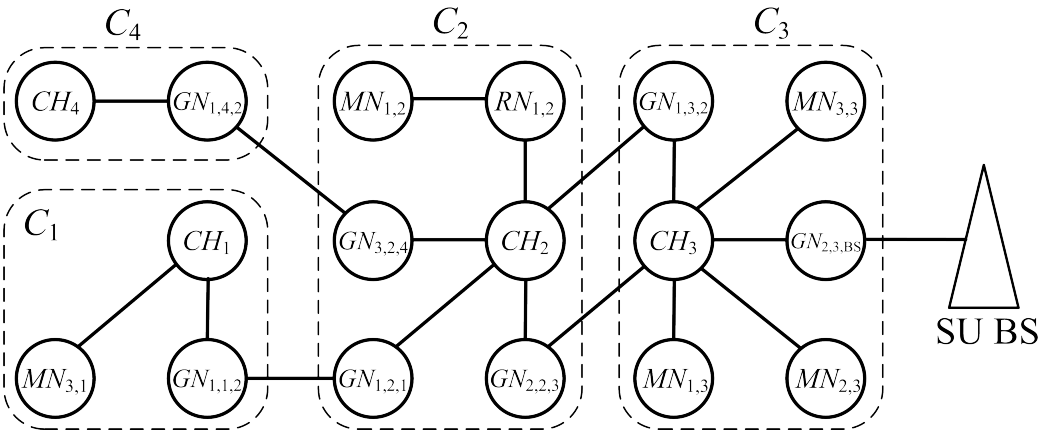

Figure 2: A cluster structure after cluster C_1 (see Figure 1) undergoes cluster splitting in a CRN. Cluster C_1 and the newly-formed cluster C_4 either use existing route or search for new routes to destination SU BS.

$$Q_n^{t+1}(s_n^t, a_n^t) \leftarrow (1-\alpha)Q_n^t(s_n^t, a_n^t) + \alpha\left[r_n^{t+1}(s_n^{t+1}) + \gamma \max_{a\in A} Q_n^t(s_n^{t+1}, a)\right] \quad (1)$$

where

- State $s_n^t \in S$ represents the decision-making factors, which affect the reward (or network performance), observed by an agent from the operating environment.
- Action $a_n^t \in A$ represents an agent's action, which may change or affect the state (or operating environment) and reward (or network performance), and so the agent learns to take optimal actions at most of the times.
- Delayed reward $r_n^{t+1}(s_n^{t+1}) \in R$ represents the positive or negative effects of an agent's action on its operating environment in the previous time instance. In other words, it is the consequence of the previous action on the operating environment in the form of network performance. It is received at time $t+1$ for an action taken at time t.
- Discount factor $0 \leq \gamma \leq 1$. The higher the value of γ, the greater the agent relies on the discounted future reward $\gamma \max_{a\in A} Q_n^t(s_n^{t+1}, a)$ compared to the delayed reward $r_n^{t+1}(s_n^{t+1})$.

- Learning rate $0 \leq \alpha \leq 1$. The higher the value of α, the greater the agent relies on rewards, including the delayed reward $r_n^{t+1}(s_n^{t+1})$ and the discounted future reward $\gamma \max_{a \in A} Q_n^t(s_n^{t+1}, a)$, compared to the Q-value $Q_n^t(s_n^t, a_n^t)$ at time t.

At decision epoch t, an agent n observes its operating environment to determine its current state s_n^t. Based on the state s_n^t, the agent chooses an action a_n^t. Next, at decision epoch $t+1$, the state s_n^t changes to s_n^{t+1} as a consequence of the action a_n^t, and the agent receives delayed reward $r_n^{t+1}(s_n^{t+1})$. Subsequently, the Q-value $Q_n^{t+1}(s_n^t, a_n^t)$ is updated using Equation (1). Note that, in the remaining decision epochs at time $t, t+1, ...$, the agent is expected to take optimal actions with regard to the states; hence Q-value is updated using a maximized discounted future reward $\gamma \max_{a \in A} Q_n^t(s_n^{t+1}, a)$. As this procedure evolves through time, the agent n receives a sequence of rewards and the Q-values converge.

RL has been applied to routing [8, 9], and its advantages are as follows:

- Instead of tackling every single factor that affects the network performance, RL models the system performance that covers a wide range of factors in the operating environment or network conditions affecting the network performance (i.e., the channel utilization level by PUs and channel quality); hence, it is a simple modeling approach.

- Prior knowledge of the operating environment or network conditions is not necessary; and so a SU can learn about the operating environment as time goes by.

3. Related Work

This section presents related work on clustering algorithms and cluster-based routing schemes in CRNs.

3.1. Clustering Algorithms

Various clustering algorithms are based on graph domination principles [7, 10]. The graph domination principle selects a small dominating set of nodes to serve as clusterheads and the rest of the nodes associate themselves with the clusterheads and become their member nodes.It has been shown that finding the minimum dominating set in distributed networks [11] and selection of a common channel [12] are both NP-hard problems. Hence, this

article proposes heuristic algorithms for SMART, which is a cluster-based routing scheme. In SMART, clustering algorithm aims to form clusters that fulfill the requirements on the number of common channels in a cluster (see C.2 in Section 1) and allow nodes to broadcast routing control messages efficiently without broadcasting on all the available channels (see C.3 in Section 1); while routing algorithm aims to find a route that maximizes the utilization of white spaces (see C.1 in Section 1) in order to maximize SUs' network performance. The route serves as a backbone throughout the network, and it is comprised of clusterheads and gateway nodes.

Most clustering schemes have been performed in two main ways, which we call *cluster-first* and *clusterhead-first* approaches. The cluster-first approach forms clusters, and subsequently a clusterhead is selected in each cluster while the rest of the nodes in the cluster become member nodes. For instance, in [6, 13], nodes with common channels or that are geographically close form a cluster. Subsequently, a node with favorable characteristics (or dominating node) in the cluster, such as greater channel sensing capability, is selected to serve as clusterhead [6]; alternatively, nodes take equal opportunity to serve as clusterheads [13]. Hence, in cluster merging, it is necessary to combine nodes from two clusters into a single cluster, and subsequently to relinquish a clusterhead and to select a common channel for the newly merged cluster. Likewise, in cluster splitting, it is necessary to separate nodes into two clusters, and subsequently to select a new clusterhead and to select a common channel for the newly split cluster.

The clusterhead-first approach achieves the effects of cluster merging and cluster splitting through the formation and disappearance of a clusterhead while the rest of the nodes associate themselves with the clusterhead. For instance, in [14], each node determines its suitability to serve as a clusterhead among its neighboring nodes. Subsequently, the suitable nodes (or dominating nodes) elect themselves as clusterheads while the rest of the nodes associate themselves with one of the neighboring clusterheads.

SMART adopts the clusterhead-first approach for cluster formation and cluster merging, and the cluster-first approach for cluster splitting. This work provides extensions through cluster merging and splitting which have not been investigated in the context of CRNs. In cluster merging, a cluster is reduced from the network. This means that clusterheads relinquish their role. Subsequently, a new clusterhead is selected and the member nodes of two clusters join the newly created cluster. In cluster splitting, a new cluster is formed in the network. This means that an existing member node

of a cluster is selected to serve as clusterhead and a common channel is selected. Subsequently, existing member nodes of the cluster may join this newly-formed cluster.

3.2. Cluster-based Routing Algorithms

While there have been a large number of separate investigations into clustering [15, 16] and routing [17–19], there is only a perfunctory attempt to investigate cluster-based routing schemes in CRNs.

Huang et al. [14] propose a cluster-based routing scheme. The cluster formation procedure adopts the clusterhead-first approach (see Section 3.1) and uses several clustering metrics, namely the node degree level (or the number of neighbor nodes), the average number of hops in a cluster, and the average number of channel switches due to distinctive channels being selected by a node and its neighbor nodes. Particle swarm optimization is applied to enable clusterheads to select common channels for inter-cluster communications. Subsequently, routing is performed to select a route with the highest availability probability, which is the product of link availability probability along the route. This approach has been shown to achieve higher throughput, lower number of clusters in the network, and lower end-to-end delay.

Talay and Altilar [20] propose another cluster-based routing scheme. The cluster formation procedure adopts the cluster-first approach (see Section 3.1). Firstly, the cluster formation procedure uses several clustering metrics, namely a set of available channels, physical location, movement, speed and moving direction of each node to select nodes for a cluster. Secondly, a clusterhead is selected based on a weighted clustering metric that takes into account the node degree and mobility levels, as well as the set of available channels of each node. The number of member nodes in a cluster must not exceed a pre-defined value. Subsequently, routing is performed to select a route that increases connectivity (or reduces the effects from PUs' activities) and reduces interference levels among SUs (i.e., reduces signal to interference and noise ratio (SINR) and expected transmission time (ETT)). This approach has been shown to achieve higher packet delivery ratio, higher throughput, lower end-to-end delay, and lower routing overhead.

SMART formulates and solves the cluster-based routing scheme using RL, and it provides further enhancements to [14, 20] in two main aspects, particularly cluster size adjustment and cluster maintenance. Cluster size adjustment changes the cluster size based on network performance while

fulfilling the requirements on the number of common channels in a cluster; while cluster maintenance is comprised of cluster merging and splitting.

4. System Model

We consider a distributed CRN. Each SU must minimize interference to PUs, and so performs CR functions, including channel sensing, channel selection, channel sharing and channel hand-off in a distributed manner. There are $n \in N = \{1, 2, 3, ..., |N|\}$ SUs, $j \in J = \{1, 2, 3, ..., |J|\}$ PUs and $k \in K = \{1, 2, 3, ..., |K|\}$ channels. $J_k \subseteq J$ represents a set of PUs $j \in J_k$ that use channel k.

The rest of this section presents our CRN architecture comprised of internal and external environments.

4.1. CRN Architecture

We introduce CR functions incorporated in QualNet. The original version of QualNet is lack of CR environment, so we introduce an architecture shown in Figure 3, and it is comprised of the internal and external environments of a SU. The rest of this section provides descriptions of each component which has been implemented in our simulation platform QualNet so that the research can focus on cluster-based routing, rather than the underlying CR functions, such as channel sensing and channel sharing.

4.1.1. Internal Environment

The components in the internal environment can be categorized into three main layers (i.e., physical, data link and network layers) and a cross-layer repository. The physical layer includes channel hand-off. The data link layer includes channel sensing and channel sharing. The network layer includes channel decision and routing protocol.

4.1.1.1. Data Link Layer

. Each SU has a single network interface which is used for data and control packet transmissions. There are two main components in the data link layer, namely channel sensing and channel sharing. *Channel sensing* module detects white spaces and determines the channel utilization level by PUs in the sensing channels through interacting with PU activity module (see Section 4.1.2.2). *Channel sharing* module enables distributed channel access among SUs in a shared wireless environment. Medium access control (MAC) protocol, namely IEEE 802.11, is applied to coordinate transmissions among

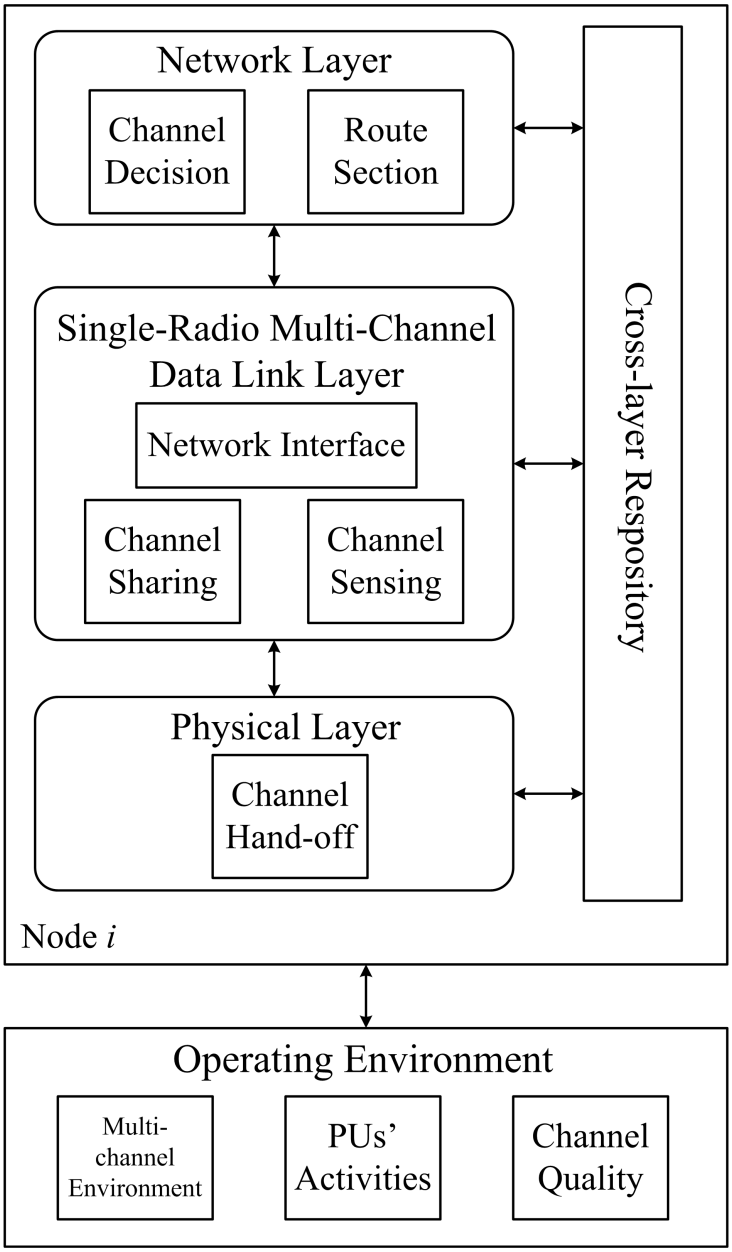

Figure 3: CRN architecture.

SUs in a particular channel. Additionally, this module interacts with the PU activity module (see Section 4.1.2.2) so that the SUs' transmissions do not

interfere with PUs' activities.

4.1.1.2. Network Layer
. The network layer contains a cluster-based routing protocol called SMART, which is a joint channel decision and route selection protocol. The routing selection is described extensively in Section 6. Channel decision module receives information about white spaces from the underlying channel sensing module in the data link layer. Upon the detection of PUs' activities, this module switches to another channel, so that interruption to transmission is minimized. During a channel switch, the channel decision module enables a SU to select one of the available channels, and this is accomplished through sending an indication signal to the underlying channel hand-off module in the physical layer (see Section 4.1.1.3) in order to initiate a channel switch to a new channel. Basically, there are two main functions offered by the channel decision module, namely channel selection and channel switching. *Channel selection* enables a SU to select one of the available channels for transmission while fulfilling the channel selection criteria. This research applies a joint channel and route selection approach which has been shown to increase route stability and enhance network performance [1]. *Channel switching* enables a SU to cease transmission in a channel and switch its transmission to another available channel with white spaces, which is determined by the channel selection module. In the case of a channel switch, the channel decision module sends an indication signal, which consists of the next operating channel, to the channel hand-off module in order to initiate a channel switch.

4.1.1.3. Physical Layer
. There is a main component in the physical layer, namely channel-handoff. Channel hand-off module enables a SU to vacate its current operating channel and switch to another one. This module is invoked by an indication signal, which consists of the next operating channel, from the channel decision module in the network layer (see Section 4.1.1.2). Any transmission must be stopped during a hand-off operation.

4.1.1.4. Cross-Layer Repository
. The cross-layer repository enables different layers of protocol stack, particularly the physical, data link and network layers, to share information. Examples of information include the next operating channel from the channel decision module in network layer to the channel handoff module in physical layer, PU utilization level from the channel sensing module in data link layer

to the route selection module in network layer, as well as list of neighbor nodes and available channels at a node.

4.1.2. External Environment

The external environment is characterized by multi-channel environment, PU activity and channel quality.

4.1.2.1. Multi-channel Environment

. The operating environment consists of a list of licensed channels. The SUs exploit the white spaces in these channels in an opportunistic manner. Note that, for each link between a SU node pair, the data channels have the same amount of link capacity, although they may have different amount of white spaces.

4.1.2.2. PUs' Activities

. The PUs' activities module generates PUs' traffic in each channel according to a PUs' traffic model. The PUs' activities are independent and identically distributed across the available channels. For each channel, a PU activity is a sequence of ON-OFF periods. The arrival time is the beginning of an ON (or non-white space) period; while the departure time is the beginning of an OFF (or white space) period. The ON-OFF transitions of PU activity for PU $j \in J_k$ using channel k follows a Poisson model in which ON and OFF periods, namely $T^k_{ON,j}$ and $T^k_{OFF,j}$, are exponentially distributed with rates $\lambda^k_{ON,j}$ and $\lambda^k_{OFF,j}$, respectively [21, 22]. The mean values of exponential distribution for ON and OFF periods are given by $E[T^k_{ON,j}] = 1/\lambda^k_{ON,j}$ and $E[T^k_{OFF,j}] = 1/\lambda^k_{OFF,j}$, respectively. We assume that all PUs' activities in a channel k are represented by T^k_{ON} and T^k_{OFF}, and each PU j uses a single channel k only [23–25].

In [25], the values of rates $\lambda^k_{ON,j}$ and $\lambda^k_{OFF,j}$ for $|K| = 10$ licensed channels have been measured by collecting samples of channel state transitions, and they are shown in Table 1. There are four kinds of PUs' activities with different rate values $\lambda^k_{ON,j}$ and $\lambda^k_{OFF,j}$ [26], which are also investigated in Section 7.3.1:

a. Channel with long-term PUs' activities has long ON and long OFF periods, specifically $\lambda^k_{ON,j} \leq 1$ and $\lambda^k_{OFF,j} \leq 1$, respectively. This type of channel has been observed whenever each call is likely to take a long period of time with long breaks in between. There are five such channels in Table 1, namely channels 3, 4, 5, 7 and 10.

Table 1: Exponential distribution parameter values for $|K| = 10$ licensed channels.

Rate	Channel									
	1	2	3	4	5	6	7	8	9	10
$\lambda^k_{ON,j}$	1.25	0.4	1	0.4	0.5	2	1	0.18	0.5	0.67
$\lambda^k_{OFF,j}$	0.67	2	1	0.33	1	0.29	0.25	2	1.33	0.5

b. Channel with high PUs' activity level has long ON and short OFF periods, specifically $\lambda^k_{ON,j} \leq 1$ and $\lambda^k_{OFF,j} > 1$, respectively. This type of channel has been observed in crowded areas particularly during on-peak hours and major events when the number of calls is high and each call is likely to take a long period of time. There are three such channels in Table 1, namely channels 2, 8 and 9.

c. Channel with low PUs' activity level has short ON and long OFF periods, specifically $\lambda^k_{ON,j} > 1$ and $\lambda^k_{OFF,j} \leq 1$, respectively. This type of channel has been observed during off-peak hours such as night time when the number of calls is low. There are two such channels in Table 1, namely channels 1 and 6.

d. Channel with intermittent PUs' activity level has short ON and short OFF periods, specifically $\lambda^k_{ON,j} > 1$ and $\lambda^k_{OFF,j} > 1$, respectively. This type of channel has been observed among commuters who are likely to access Internet for a short period of time. Due to the instability introduced by such channels to SUs' communication, this kind of channel is less likely to be exploited by SUs, and so it is excluded from Table 1. Nevertheless, we investigate this type of PU activity in Section 7.3.1.

4.1.2.3. Channel Quality
. Channels in CRNs are heterogeneous in nature depending on the amount of white spaces (or PUs' activities). The channel quality indicates the amount of white spaces in a channel in terms of the probability of a channel to remain in OFF-state in the next time slot calculated using channel capacity metric (see Equation (2)).

5. SMART: Clustering

The clustering in SMART consists of channel capacity metric, cluster formation, gateway node selection, cluster merging and cluster splitting. In this

section, we present SMART clustering algorithms. For clarity and simplicity, SMART is presented using three separate examples. Section 5.5.1 presents cluster merging example. Section 5.6.1 presents cluster splitting example. Section 6.1 presents routing example. These examples are relevant to each other in a sense that cluster merging and splitting are cluster maintenance mechanisms. Subsequently, routing is performed after cluster maintenance.

5.1. Channel Capacity Metric

The channel capacity metric is based on maximum likelihood estimation (MLE), which is defined as the probability of a channel being in the OFF state at time t. In SMART, channel capacity metric is used to rank the available channels in clustering and routing. The metric is as follows [25, 27]:

$$\varphi_k^t = \frac{\lambda_{ON,j}^k}{\lambda_{ON,j}^k + \lambda_{OFF,j}^k} + \frac{\lambda_{OFF,j}^k}{\lambda_{ON,j}^k + \lambda_{OFF,j}^k} e^{-(\lambda_{ON,j}^k + \lambda_{OFF,j}^k)^t} \tag{2}$$

A channel k of PU j with higher probability φ_k^t at time t has higher amount of white spaces, and so it is given a higher rank. Therefore, in SMART, channel capacity metric is applied in clustering and routing mechanisms in order to maximize the utilization of white spaces (see C.1 in Section 1).

5.2. Packet Structure

Each node (i.e., non-clustered node, clusterhead and member node) exchanges clustering message `CHinfo` among themselves. The `CHinfo` is embedded in a `Hello` message, and so it is exchanged periodically or when necessary during cluster formation, cluster merging and cluster splitting. In addition to `CH` (node ID of a new clusterhead), `merge` (`merge` = 1 indicates cluster merging is initiated), and `split` (`split` = 1 indicates cluster splitting is initiated) in clustering message `CHinfo`, other clustering information is included so that the clusterhead can calculate clustering metric and make decision on cluster merging, cluster splitting, the selection of a new clusterhead, and the relinquishment of an existing clusterhead. Note that, only clusterheads can set values for `CH`, `merge` and `split`, although all nodes (including the clusterheads) include the rest of the information in `CHinfo`. As an example, an existing clusterhead indicates the node ID of a new clusterhead (or the new cluster ID) using `CHinfo.CH`. As another example, the clustering information `CHinfo.merge` = 1 indicates that the clusterhead

Table 2: Notations used in clustering.

Notation	Description
`CHinfo`	Clustering information set by a node
`CHinfo.CH`	Identification of a new clusterhead
`CHinfo.merge`	Indication of cluster merging
`CHinfo.split`	Indication of cluster splitting
`CH`	Clusterhead of a cluster
`nodeID`	Identification of a node
`nodeState`	State of a node: non-clustered (NN), clusterhead (CH), or member node (MN)
`clusterID`	Identification of a cluster
`listChannels`	List of available channels at a node
`listNodes`	List of nodes in a cluster
`commonChannels`	List of available common channels in a cluster
`clusterSize`	Number of nodes in a cluster
`masterChannel`	Common operating channel of a cluster
`backupChannel`	Backup common channel of a cluster

initiates cluster merging, while `CHinfo.merge` = 0 indicates that it does not initiate cluster merging. Similar explanation applies to `CHinfo.split`.

Two types of the clustering information are *identification* and *cluster formation*. The identification information of a clustered node (i.e., clusterhead and member nodes) is `nodeID` (node ID), `nodeState` (the state of a node including clusterhead or member node), `clusterID` (cluster ID), `listChannels` (the list of available channels), `listNodes` (the list of nodes in a cluster) and `commonChannels` (the list of available common channels in a cluster); while identification information of a non-clustered node is `nodeID`, `nodeState` and `listChannels`. The cluster formation information, which is broadcast by clustered nodes, is `clusterSize` (the number of nodes in a cluster), `listChannels`, `masterChannel` (the common operating channel of a cluster), `backupChannel` (a backup channel to be used when the common operating channel of a cluster becomes unavailable), `listNodes` and `commonChannels`.

Notations used in clustering along with their descriptions are summarized in Table 2. A summary of messages, including notations and descriptions, used to coordinate clusters are presented in Table 3. Table 4 describes the notations used in clustering algorithms.

Table 3: Notations of messages used in clustering mechanisms.

Category	Notation	Description
Cluster joining	$JREQ_{i,j}$	Cluster joining request message from node i to neighboring clusterhead j
	$JACC_{i,j}$	Cluster joining acceptance message from neighboring clusterhead j to node i
	$JDEC_{i,j}$	Cluster joining decline message from neighboring clusterhead j to node i
	$JREC_{i,j,k}$	Cluster joining recommendation message from own clusterhead j for recommending its member node i to join neighboring clusterhead k
Gateway node selection	$GREQ_{i,j,k}$	Gateway role request message from member node i to its own clusterhead j for neighboring cluster k
	$GACC_{i,j,k}$	Gateway role acceptance message to member node i from its own clusterhead j for neighboring cluster k
	$GDEC_{i,j,k}$	Gateway role decline message to member node i from its own clusterhead j for neighboring cluster k
Cluster merging	$MREQ_{i,j,k}$	Cluster merging request message from gateway node i to its own clusterhead j and neighboring clusterhead k
	$MACC_{i,j,k}$	Cluster merging acceptance message to gateway node i from clusterhead j for merging with cluster k
	$MDEC_{i,j,k}$	Cluster merging decline message to gateway node i from clusterhead j for merging with cluster k
	$MCAN_{i,j,k}$	Cluster merging cancellation message from gateway node i to its own clusterhead j and neighboring cluster k
	$REL_{j,k}$	Clusterhead role relinquishment message from a new clusterhead k (or existing gateway node) to an existing clusterhead j
Relay node selection	$LREQ_{i,j,k}$	Relay request message from requesting member node i to another member node j for its clusterhead k
	$LACC_{i,j,k}$	Relay acceptance message from member node j to requesting member node i for its clusterhead k
	$LINF_{i,j,k}$	Relay notification message from relay node j to its clusterhead k about requesting member node i
Timers	$T_{w,scan}$	Scanning interval of a channel for receiving `CHinfo` from neighboring nodes
	$T_{w,res}$	Waiting interval for a response from a clusterhead
	$T_{w,CHE}$	Waiting interval for clusterhead election

Table 4: Notations used in clustering algorithms.

Notation	Description
NN_i	Non-clustered node i
CH_j	Clusterhead j
$MN_{i,j}$	Member node i of clusterhead j
$GN_{i,j,k}$	Gateway node i of clusterhead j that provides inter-cluster connectivity between its own clusterhead and neighboring clusterhead k
$RN_{i,j}$	Relay node i that provides intra-cluster connectivity between member node j and its own clusterhead
$N_{\texttt{CHinfo.nodeState=CH}}$	Number of clustering message `CHinfo` received from clusterheads
MC_{CH_i}	Master channel of clusterhead i
NB_i	SU neighbor nodes of SU i
$NN_{l \in NB_i}$	Non-clustered SU neighbor nodes of SU i
$N_{NN_{l \in NB_i}}$	Number of non-clustered SU neighbor nodes of SU i
$N_{N \in C_i,k}$	Number of nodes that belongs to cluster C_i, having channel k in their list of available channels
$\gamma^t_{CH,j}$	Rank of clusterhead j
$\gamma^t_{chan,k}$	Rank of channel k
n_{c,C_j}	Number of common channels in a cluster C_j
$n_{c,C_{i,j}}$	Number of common channels among clusters C_i and C_j
$H_{C,min}$	Threshold for the minimum number of common channels in a cluster
$H_{C,merge}$	Cluster merging threshold for the minimum number of common channels required for cluster merging
φ^t_k	Channel capacity for channel k
$N_{listChannels_i}$	Number of available channels of SU i
H_{i,C_l}	Number of hops between SU i and neighboring cluster l

5.3. Cluster Formation

All SUs are in non-clustered state at the initial stage. Cluster formation creates logical groups (or clusters) consists of clusterheads and member nodes. Initially, each SU scans each of the available channels for a short time duration $T_{w,scan}$ during which a node may receive clustering message `CHinfo` from its neighboring nodes (e.g., clustered and non-clustered nodes), and maintain its neighbor table. Algorithm 1(a) presents cluster formation procedure at non-clustered node NN_i and it is explained in Sections 5.3.1

and 5.3.2.

5.3.1. Node Joining

Node joining is the process of associating a non-clustered node with a cluster. SMART fulfils the availability of a certain number of common channels in a cluster upon node joining in order to maximize stability (see C.2 in Section 1). The node joining process is illustrated in Algorithms 1(a) and 1(b). In part I of Algorithm 1(a), a SU i scans the list of available channels in a sequential manner, and each channel is scanned for $T_{w,scan}$ duration. Upon scanning all the available channels in the list $listChannels_i$, if a SU receives `CHinfo`, it stores the sender of `CHinfo` and the respective information in its neighbor table.

In part II of Algorithm 1(a), there are two circumstances in which a SU chooses to join a clusterhead. Firstly, a SU has received clustering message `CHinfo` from a single clusterhead (i.e., $N_{\texttt{CHinfo.nodeState=CH}} = 1$), and so this clusterhead is chosen. Secondly, a SU has received more than one clustering message `CHinfo` from different clusterheads (i.e., $N_{\texttt{CHinfo.nodeState=CH}} > 1$), then it ranks the master channels of the clusterheads based on channel capacity metric φ_k^t (see Equation (2) in Section 5.1).

The clusterheads are ranked such that a clusterhead j has the highest rank (i.e., $\gamma_{CH,j}^t = 1$) if its master channel k has the highest channel capacity among the channel capacities of master channels of other neighboring clusterheads (i.e., $\varphi_k^t > \varphi_{l \in MC_{CH_{m \in NB_i}}}^t$). Similarly, other clusterheads are ranked as second, third and so on. Finally, the node selects a clusterhead j with the highest rank (i.e., $\gamma_{CH,j}^t = 1$).

Next, in both circumstances, a SU i sends a cluster joining request ($JREQ_{i,j}$) to the selected clusterhead j, and waits for a response from the clusterhead j within a time duration $T_{w,res}$. If SU i receives an acceptance response ($JACC_{i,j}$) from clusterhead j, it becomes the member node $MN_{i,j}$ of the respective cluster j (i.e., $\texttt{nodeState}_i \leftarrow MN_{i,j}$); otherwise, the next clusterhead with the highest rank using its master channel is chosen.

Next, we focus on the circumstance in which a clusterhead CH_j receives $JREQ_{i,j}$ message from a non-clustered node i, as shown in Algorithm 1(b). The clusterhead CH_j only accepts a joining request by sending back cluster joining accept ($JACC_{i,j}$) message if the number of common channels n_{c,C_j} in its cluster C_j fulfils the threshold for the minimum number of common channels in a cluster (i.e., $n_{c,C_j} \geq H_{C,min}$) upon node joining in order to maximize cluster stability (see C.2 in Section 1). Otherwise, it declines the

joining request by sending back cluster joining decline ($JDEC_{i,j}$) message to the SU i.

5.3.2. Clusterhead Election

A clusterhead serves as a local point of process for all member nodes in its cluster, and it is responsible for coordinating tasks among its member nodes. For example, it collects channel sensing outcomes from its member nodes and makes the final decision on channel availability. SMART uses a clustering metric that elects a node with the highest number of available channels as clusterhead during clusterhead election in order to avoid frequent re-election. A clusterhead also processes cluster joining request from non-clustered nodes. In SMART, a clusterhead must ensure that its cluster fulfils the requirement of the minimum number of common channels upon any new node joining in order to maximize cluster stability (see C.2 in Section 1). In part III of Algorithm 1(a), a SU i may not receive clustering message `CHinfo` from any clusterhead if there is lack of clusterhead in its neighborhood, and so it remains in non-clustered state. It starts to form a cluster with non-clustered SU neighbor nodes $NN_{l \in NB_i}$.

There are two circumstances. Firstly, there is lack of non-clustered SU neighbor nodes of SU i (i.e., $N_{NN_{l \in NB_i}} = 0$), and so SU i forms a cluster itself and becomes a clusterhead (i.e., $\texttt{nodeState}_i \leftarrow CH_i$). Secondly, there is at least a single non-clustered SU neighbor node (i.e., $N_{NN_{l \in NB_i}} \geq 1$), and so SU i becomes a clusterhead if it has the highest clustering metric, specifically the highest number of available channels ($N_{listChannels_i} \geq N_{listChannels_{j \in NN_{l \in NB_i}}}$), among its non-clustered SU neighbor nodes. Subsequently, the new clusterhead ranks its available channels $listChannels_i$ using channel capacity metric φ_k^t (see Equation (2) in Section 5.1) and selects a master channel with the highest rank $\gamma_{chan,k}^t = 1$, and a backup channel with the second highest rank $\gamma_{chan,k}^t = 2$; and subsequently broadcast this information using clustering message `CHinfo`. However, if SU i does not have the highest clustering metric among its non-clustered SU neighbor nodes, it sets a timer $T_{w,CHE}$ to allow non-clustered SU neighbor nodes with the highest clustering metric among the respective neighborhood to become clusterhead and joins the cluster with the highest rank. Note that, if SU does not receive any clustering message `CHinfo` from any clusterhead upon the expiration of the timer $T_{w,CHE}$, it starts another round of process for non-clustered node.

Algorithm 1(a) Cluster formation procedure at non-clustered node NN_i

1: /* *Part I: Scan each available channel in order to receive clustering message* `CHinfo` */
2: **while** $listChannels_i$ **do**
3: Scan each available channel k for $T_{w,scan}$ duration;
4: **if** receive `CHinfo` **then**
5: Store `CHinfo`;
6: **end if**
7: **end while**
8: /* *Part II: Process* `CHinfo` *received from clusterhead(s)* */
9: **if** $N_{\texttt{CHInfo.nodestate=}CH} = 1$ **then**
10: Send $JREQ_{i,j}$ to CH_j;
11: **else if** $N_{\texttt{CHInfo.nodestate=}CH} > 1$ **then**
12: **for** k in `CHinfo.nodeID.masterChannel` **do**
13: Calculate φ_k^t using Equation (2)
14: **end for**
15: Update $\gamma_{CH,j\in NB_i}^t$ such that $\gamma_{CH,j}^t > \gamma_{CH,l\in NB_i}^t$ if $\phi_{MC_j}^t > \phi_{MC_l}^t$;
16: **while** not receive $JACC_{i,j}$ or $CH_{l\in\texttt{CHinfo.nodeState=}CH} = \Phi$ **do**
17: Send $JREQ_{i,j}$ to $CH_j | \gamma_{CH,j}^t > \gamma_{CH,l\in NB_i}^t$;
18: Wait $T_{w,res}$;
19: **if** receive $JACC_{i,j}$ from CH_j **then**
20: $\texttt{nodeState}_i \leftarrow MN_{i,j}$;
21: break;
22: **end if**
23: **end while**
24: /* *Part III: Process* `CHinfo` *received from non-clustered node(s)* */
25: **else if** $N_{NN_{l\in NB_i}} = 0$ or $N_{listChannels_i} > N_{listChannels_{j\in NN_{l\in NB_i}}}$ **then**
26: $\texttt{nodeState}_i \leftarrow CH_i$;
27: **for** k in $listChannels_i$ **do**
28: Calculate φ_k^t using Equation (2)
29: **end for**
30: Update $\gamma_{chan,k}^t$ such that $\gamma_{chan,k}^t > \gamma_{chan,m}^t$ if $\varphi_k^t > \varphi_m^t | k \in listChannels_i$ and $m \in listChannels_i$;
31: $\texttt{masterChannel} = k | \gamma_{chan,k}^t = 1$;
32: $\texttt{backupChannel} = k | \gamma_{chan,k}^t = 2$;
33: Broadcast `CHinfo`;
34: **else**
35: Wait $T_{w,CHE}$;
36: **if** receive `CHinfo` from CH_j **then**
37: Send $JREQ_{i,j}$ to CH_j;
38: **else**
39: Run Algorithm 1(a);
40: **end if**
41: **end if**

Algorithm 1(b) Cluster formation procedure at clusterhead CH_j

1: Receive $JREQ_{i,j}$ from non-clustered node i;
2: **if** $n_{c,C_j} \geq H_{C,min}$ after node i joins cluster C_j **then**
3: Send $JACC_{i,j}$;
4: **else**
5: Send $JDEC_{i,j}$;
6: **end if**

5.4. Gateway Node Selection

In CRNs, adjacent clusters may operate on different channels due to the availability of multiple channels in the network, and so a single gateway node is insufficient to provide two-way inter-cluster communications. Specifically, given that a clusterhead does not switch from its master channel, a two-way inter-cluster communication may require each cluster to select a distinct gateway node in order to send packets between the clusters. There are two cases depending on the number of hops between clusterheads in the adjacent clusters: (a) two hops, and (b) more than two hops. These situations are best explained using examples.

We first present the first case in which adjacent clusterheads communicate with each other in two hops using a single gateway node. Figure 4 shows that a clusterhead CH_1 in cluster C_1 sends a data packet to a neighboring cluster C_2 operating on a different master channel. Firstly, clusterhead CH_1 forwards data packets to its gateway node $GN_{1,1,2}$ using its master channel. Secondly, gateway node $GN_{1,1,2}$ switches to the master channel of neighboring cluster C_2 and forwards data packets to clusterhead CH_2. Thirdly, upon completion of data packets transmissions, gateway node $GN_{1,1,2}$ switches back to the master channel of its own cluster C_1. Suppose, clusterhead CH_2 wants to send data packets to clusterhead CH_1, it cannot forward data packets to gateway node $GN_{1,1,2}$ unless it first switches to the master channel of cluster C_1. However, clusterhead CH_2 cannot switch from the master channel of its own cluster. Hence, clusterhead CH_2 selects gateway node $GN_{1,2,1}$, which can switch its operating channel to the master channel of cluster C_1, in order to forward data packets to clusterhead CH_1.

Next, we present the second case in which adjacent clusterheads communicate with each other in more than two hops using more than a single gateway node. A set of gateway nodes connecting two clusterheads is called joint gateway nodes [28, 29]. Figure 5 illustrates gateway nodes (i.e., $GN_{1,1,2}$

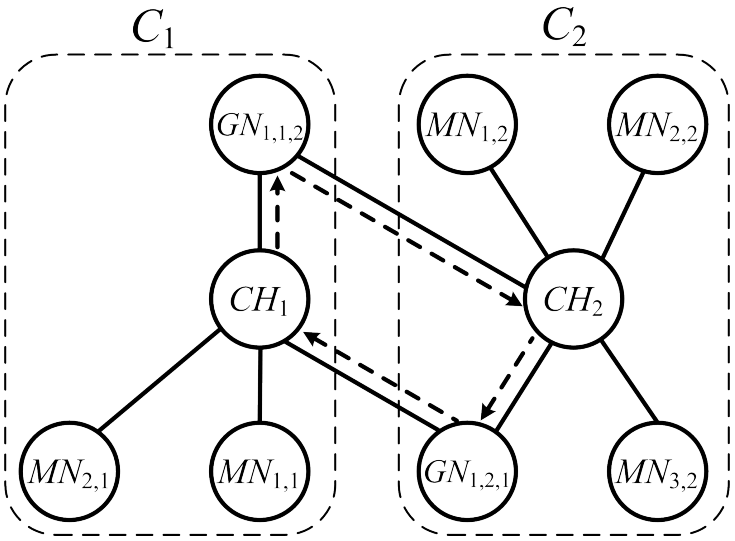

Figure 4: Gateway nodes forward data packets to neighboring clusters.

and $GN_{1,2,1}$) in the first case and a set of joint gateway nodes (i.e., $GN_{2,2,3}$ and $GN_{1,3,2}$) in the second case. Using joint gateway nodes, a two-way inter-cluster communication uses a similar set of gateway nodes.

Next, we present the procedures for gateway node selection which are presented in Algorithms 2(a) and 2(b) at member node $MN_{i,j}$ and clusterhead CH_j, respectively. In Algorithm 2(a), a member node $MN_{i,j}$ periodically scans each of its available channels $listChannels_{MN_{i,j}}$ for $T_{w,scan}$ duration within each time window in order to discover neighboring clusters. Suppose, it receives clustering message `CHinfo`, which consists of `clusterID`, `masterChannel`, `backupChannel` and `commonChannels` from a neighboring cluster C_l. Then, it sends gateway role request message ($GREQ_{i,j,l}$) to its clusterhead CH_j in order to inform its own clusterhead about its potential role as a gateway node for cluster C_l. It may be possible that a clusterhead already has a gateway node to a neighboring cluster. However, SMART enables a clusterhead to explore other potential gateway nodes which may have lower number of hops leading to neighboring clusterhead and higher number of available channels. As shown in Algorithm 2(b), the clusterhead CH_j then informs its member node $MN_{i,j}$ to serve as a gateway node $GN_{i,j,l}$ to the respective neighboring cluster C_l by sending gateway role acceptance message ($GACC_{i,j,l}$), if there is lack of a gateway node to cluster C_l (i.e.,

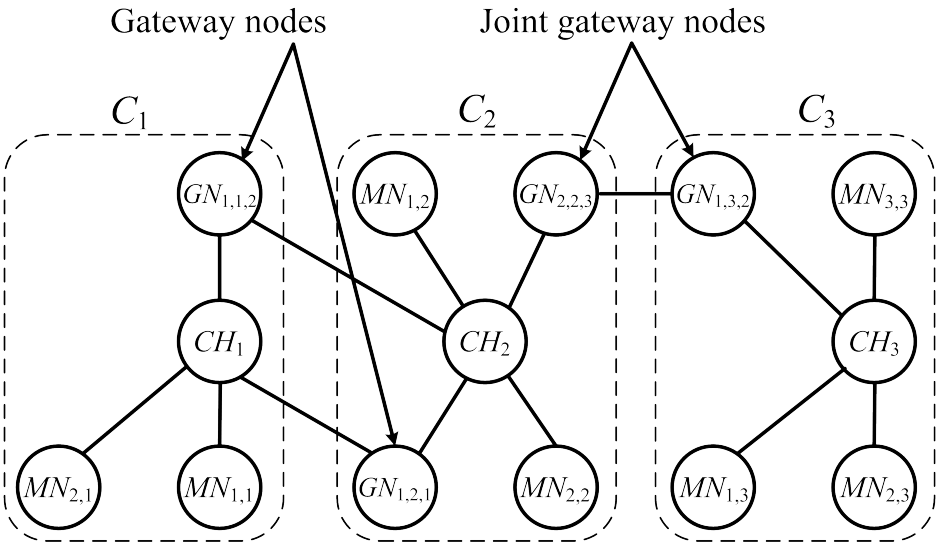

Figure 5: Gateway nodes and joint gateway nodes.

$GN_{m\in C_j,j,l} = \varnothing$) or the member node $MN_{i,j}$ has the least number of hops leading to the clusterhead CH_l of the neighboring cluster C_l as compared to existing gateway node $GN_{m,j,l}$ (i.e., $H_{MN_{i,j},C_l} < H_{GN_{m,j,l},C_l}$). If there are potential and existing gateway nodes with similar number of hops leading to the clusterhead of the neighboring cluster, the member node $MN_{i,j}$ with the highest number of available channels $N_{listChannels_{MN_{i,j}}}$ is selected. Otherwise, the clusterhead CH_j declines the request by sending gateway role decline message ($GDEC_{i,j,l}$) to its member node $MN_{i,j}$.

Algorithm 2(a) Gateway node selection procedure at member node $MN_{i,j}$

1: **while** $listChannels_{MN_{i,j}}$ **do**
2: Scan each channel k for $T_{w,scan}$ duration;
3: **if** receive `CHinfo` from C_l **then**
4: Send $GREQ_{i,j,l}$ to clusterhead CH_j;
5: **end if**
6: **end while**
7: **if** receive $GACC_{i,j,l}$ from clusterhead CH_j **then**
8: $\texttt{nodeState}_{MN_{i,j}} \leftarrow GN_{i,j,l}$;
9: **end if**

Algorithm 2(b) Gateway node selection procedure at clusterhead CH_j

Receive $GREQ_{i,j,l}$ from $MN_{i,j}$;
if $GN_{m \in C_j,j,l} = \varnothing$ **then**
    Send $GACC_{i,j,l}$ to $MN_{i,j}$;
else if $H_{MN_{i,j},C_l} < H_{GN_{m,j,l},C_l}$ **then**
    Send $GACC_{i,j,l}$ to $MN_{i,j}$;
    Send $GDEC_{m,j,l}$ to $GN_{m,j,l}$;
else if $H_{MN_{i,j},C_l} > H_{GN_{m,j,l},C_l}$ **then**
    Send $GDEC_{i,j,l}$ to $MN_{i,j}$;
else if $H_{MN_{i,j},C_l} = H_{GN_{m,j,l},C_l}$ **then**
    if $N_{listChannels_{MN_{i,j}}} > N_{listChannels_{GN_{m,j,l}}}$ **then**
        Send $GACC_{i,j,l}$ to $MN_{i,j}$;
        Send $GDEC_{m,j,l}$ to $GN_{m,j,l}$;
    else
        Send $GDEC_{i,j,l}$ to $MN_{i,j}$;
    end if
end if

5.5. Cluster Merging

Cluster merging is the process of combining two clusters into one. In SMART, cluster merging is only possible when the number of common channels $n_{c,C_{i,j}}$ between clusters i and j satisfies a threshold for cluster merging $H_{C,merge}$, specifically the minimum number of common channels in a merged cluster, for cluster stability (see C.2 in Section 1).

5.5.1. Example

An example of cluster merging is given in Figure 6. Suppose, the cluster merging threshold is 2 (i.e., $H_{C,merge} = 2$). Gateway node $GN_{1,1,2}$ discovers the set of common channels between clusters C_1 and C_2 (i.e., {2,3}), and it fulfils the threshold $H_{C,merge}$. Thus, it informs both clusterheads CH_1 and CH_2 about the potential cluster merging activity in which it can serve as a clusterhead. Suppose, both clusterheads agree to merge, then each of them sends a positive response and their respective list of member nodes to the gateway node. Next, the gateway node $GN_{1,1,2}$ becomes clusterhead CH_1 and the existing clusterheads relinquish their roles and become member nodes of clusterhead CH_1. Since member nodes $MN_{1,1}$ and $MN_{1,2}$ are located within the transmission range of the new clusterhead CH_1, both of them become

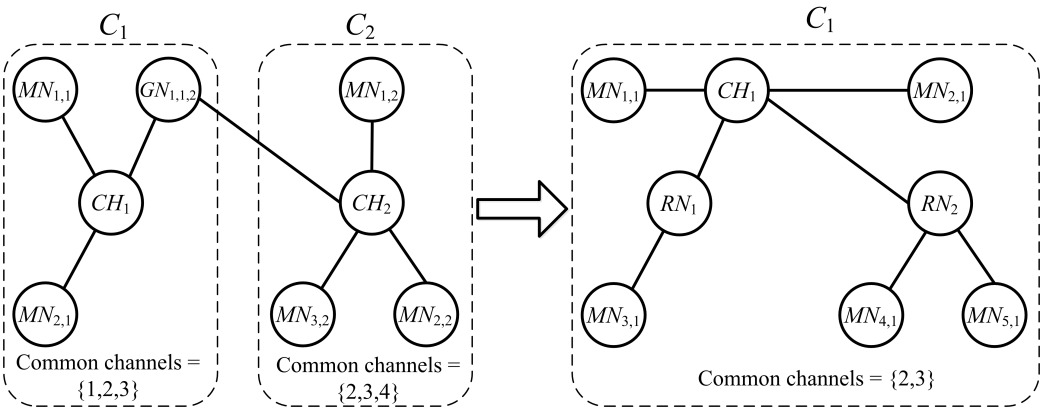

Figure 6: An example of cluster merging from clusters C_1 and C_2 to cluster C_1.

member nodes of the new clusterhead. Since member nodes $MN_{2,1}$, $MN_{2,2}$ and $MN_{3,2}$ are located out of the transmission range of the new clusterhead, the relinquished clusterheads become relay nodes (i.e., RN_1 and RN_2) for these nodes.

5.5.2. Procedure

Algorithms 3(a), 3(b) and 3(c) present the cluster merging procedure for cluster C_j at gateway node $GN_{i,j,l}$, clusterhead CH_j and member node $MN_{i,j}$, respectively.

In part I of Algorithm 3(a), a gateway node $GN_{i,j,l}$ is aware of a set of common channels $n_{c,C_{j,l}}$ between its cluster C_j and neighboring cluster C_l to which it is connected to. Thus, whenever a gateway node $GN_{i,j,l}$ discovers a potential cluster merging activity (i.e., $n_{c,C_{j,l}} \geq H_{C,merge}$), it informs both clusterheads CH_j and CH_l by sending cluster merging request message ($MREQ_{i,j,l}$). In part I of Algorithm 3(b), a clusterhead CH_j may accept the cluster merging request by sending cluster merging acceptance message ($MACC_{i,j,l}$) to $GN_{i,j,l}$ to merge clusters. This occurs when CH_j is not undergoing cluster merging with another cluster (i.e., $MREQ_{*,j,*} = \varnothing$). Subsequently, in part II of Algorithm 3(a), a gateway node $GN_{i,j,l}$ receives $MACC_{i,j,l}$ and $MACC_{i,l,j}$ from both clusterheads CH_j and CH_l respectively, and becomes the clusterhead CH_m of the newly-formed cluster C_m. After becoming a clusterhead, it selects master and backup channels, broadcast `CHinfo` and sends clusterhead role relinquishment messages $REL_{j,m}$ and $REL_{l,m}$ to CH_j and CH_l respectively, so that they relinquish their cluster-

head role and become its member nodes. Note that, one of the clusterheads, say clusterhead CH_j, may not agree to merge and send cluster merging decline message ($MDEC_{i,j,l}$) to $GN_{i,j,l}$. In this case, as shown in part II of Algorithm 3(b), the gateway node $GN_{i,j,l}$ informs both clusterheads about the cancellation of the cluster merging by sending cluster merging cancellation message ($MCAN_{i,j,l}$). This allows both clusterheads to accept cluster merging requests from other gateway nodes by ignoring the current cluster merging procedure (i.e., $MREQ_{i,j,l} \leftarrow \varnothing$).

In part III of Algorithm 3(b), upon receiving clusterhead relinquishment message $REL_{j,m}$ by existing clusterhead CH_j from new clusterhead CH_m, the clusterhead CH_j informs its member nodes $MN_{*,j}$ to join the new clusterhead CH_m on the new master channel by sending cluster joining recommendation message $\text{JREC}_{i,j,m} \forall i \in MN_{*,j}$, marks itself as member node of the new clusterhead CH_m (i.e., $\texttt{nodeState}_j \leftarrow MN_{j,m}$), and sets its clusterhead being the new one (i.e., $CH_j \leftarrow CH_m$).

In part I of Algorithm 3(c), there are two circumstances when a member node receives cluster joining recommendation message. Firstly, a member node is located in the transmission range of the new clusterhead CH_m, and so after receiving cluster joining recommendation message $JREC_{i,j,m}$ from its existing clusterhead CH_j, it sends cluster joining request $JREQ_{i,m}$ to the new clusterhead CH_m and joins its cluster upon receiving cluster joining acceptance message $JACC_{i,m}$. Secondly, the member node is located out of the transmission range of the new clusterhead CH_m, and so it does not receive cluster joining acceptance message $JACC_{i,m}$ message within $T_{w,res}$ duration. In this case, it sends relay request message $LREQ_{i,j}$ to its relinquished clusterhead CH_j (currently $MN_{j,m}$), so that it serves as a relay node to the new clusterhead CH_m.

In part II of Algorithm 3(c), the relinquished clusterhead (currently member node $MN_{j,m}$), upon receiving relay request message $LREQ_{i,j}$ from member node $MN_{i,m}$, changes its state to relay node $RN_{i,j}$, sends relay information message $LINF_{i,j,m}$ to the new clusterhead CH_m informing about its role as relay node for member node $MN_{i,m}$, and finally sends relay acceptance message ($LACC_{i,j,l}$) to member node $MN_{i,m}$.

5.6. Cluster Splitting

Cluster splitting is the process of splitting one cluster into two clusters. In SMART, cluster splitting occurs when a clusterhead CH_j finds that the number of common channels in a cluster n_{c,C_j} is below a threshold for the

Algorithm 3(a) Cluster merging procedure at gateway node $GN_{i,j,l}$

/* *Part I: Gateway node determines potential cluster merging* */
if $n_{c,C_{j,l}} \geq H_{C,merge}$ **then**
    Send $MREQ_{i,j,l}$ to CH_j and CH_l;
end if
/* *Part II: Gateway node performs or cancels cluster merging* */
if receive $MACC_{i,j,l}$ and $MACC_{i,l,j}$ from CH_j and CH_l, respectively **then**
    Create C_m;
    $\texttt{clusterID}_m \leftarrow i$;
    $CH_m \leftarrow GN_{i,j,l}$;
    $\texttt{nodeState}_i \leftarrow CH_i$;
    for k in $listChannels_i$ **do**
        Calculate φ_k^t using Equation (2)
    end for
    Update $\gamma_{chan,k}^t$ such that $\gamma_{chan,k}^t > \gamma_{chan,m}^t$ if $\varphi_k^t > \varphi_m^t \mid k \in listChannels_i, m \in listChannels_i$;
    $\texttt{masterChannel} = k \mid \gamma_{chan,k}^t = 1$;
    $\texttt{backupChannel} = k \mid \gamma_{chan,k}^t = 2$;
    Broadcast `CHinfo`;
    Send $REL_{j,m}$ and $REL_{l,m}$ to CH_j and CH_l, respectively;
else if receive $MDEC_{i,j,l}$ from CH_j or $MDEC_{i,l,j}$ from CH_l **then**
    Send $MCAN_{i,j,l}$ to CH_j and CH_l;
end if

minimum number of common channels in a cluster due to PUs' activities (i.e., $n_{C,C_j} < H_{C,min}$) for cluster stability (see C.2 in Section 1).

5.6.1. Example

An example of cluster splitting is given in Figure 7. Initially, cluster C_1 is comprised of 6 nodes with `masterChannel`=5, `backupChannel`=6 and `commonChannels`={5,6}. Suppose, channels 5 and 6 are occupied by PUs and the threshold for minimum number of common channels in a cluster is $H_{C,min} = 2$. So, there is no common channel available for intra-cluster communication (i.e., $n_{CC_{C_1}} = 0 < H_{C,min} = 2$) and thus cluster splitting takes place. The clusterhead CH_1 is aware about a list of available channels *listChannels* of all nodes in its cluster. Thus, it counts the number

Algorithm 3(b) Cluster merging procedure at clusterhead CH_j

1: */* Part I: Clusterhead makes decision for cluster merging */*
2: **if** receive $MREQ_{i,j,l}$ from $GN_{i,j,l}$ **then**
3: **if** $MREQ_{*,j,*} = \varnothing$ **then**
4: Send $MACC_{i,j,l}$ to $GN_{i,j,l}$;
5: **else**
6: Send $MDEC_{i,j,l}$ to $GN_{i,j,l}$;
7: **end if**
8: **end if**
9: */* Part II: Clusterhead receives cluster merging cancellation message */*
10: **if** receive $MCAN_{i,j,l}$ from $GN_{i,j,l}$ **then**
11: $MREQ_{i,j,l} \leftarrow \varnothing$;
12: **end if**
13: */* Part III: Existing clusterhead receives clusterhead relinquishment message from new clusterhead */*
14: **if** receive $REL_{j,m}$ from CH_m **then**
15: Send $JREC_{i,j,m} \forall\ i \in MN_{*,j}$;
16: $\texttt{nodeState}_j \leftarrow MN_{j,m}$;
17: $\texttt{ClusterID}_j \leftarrow m$;
18: $CH_j \leftarrow CH_m$;
19: **end if**

of nodes in each available channel and ranks these channels based on maximum node degree. In Figure 7, channel 1 is available to 4 nodes (i.e., CH_1, $MN_{1,1}$, $MN_{2,1}$ and $MN_{3,1}$), channel 2 is available to 5 nodes (i.e., CH_1, $MN_{1,1}$, $MN_{2,1}$, $MN_{3,1}$ and $MN_{4,1}$), channel 3 is available to 4 nodes (i.e., CH_1, $MN_{1,1}$, $MN_{2,1}$ and $MN_{3,1}$), channel 4 is available to 3 nodes (i.e., CH_1, $MN_{4,1}$ and $MN_{5,1}$), and channel 7 is available to 3 nodes (i.e., CH_1, $MN_{4,1}$ and $MN_{5,1}$). Therefore, channels are ranked as $\gamma^t_{chan,2} = 1$, $\gamma^t_{chan,1} = 2$, $\gamma^t_{chan,3} = 3$, $\gamma^t_{chan,4} = 4$ and $\gamma^t_{chan,7} = 5$. Afterwards, the channels, which fulfils $H_{C,min} = 2$ channels, are ranked first and second (i.e., channels 2 and 1) and identifies the nodes having these channels in their respective list of available channels (e.g., CH_1, $MN_{1,1}$, $MN_{2,1}$ and $MN_{3,1}$). Subsequently, it forms a cluster of these nodes (i.e., cluster C_1 in Figure 7 after cluster splitting), selects a node as clusterhead which has the highest clustering metric (i.e., the highest number of available channels in a cluster), and sends clustering message `CHinfo` containing information of the newly created cluster,

Algorithm 3(c) Cluster merging procedure at member node $MN_{i,j}$

1: */* Part I: Member node receives new cluster joining recommendation from existing clusterhead*/*
2: **if** receive $JREC_{i,j,m}$ from CH_j **then**
3: Send $JREQ_{i,m}$ to CH_m;
4: **if** not receive $JACC_{i,m}$ within $T_{w,res}$ from CH_m **then**
5: Send $LREQ_{i,j,m}$ to CH_j;
6: **else**
7: $\texttt{nodeState}_i \leftarrow MN_{i,m}$;
8: $\texttt{ClusterID}_i \leftarrow m$;
9: $CH_i \leftarrow CH_m$;
10: **end if**
11: **end if**
12: */* Part II: Relinquished clusterhead (i.e., currently member node $MN_{j,m}$) receives relay request from member node $MN_{i,m}$ */*
13: **if** receive $\text{LREQ}_{i,j,m}$ from $MN_{i,m}$ **then**
14: $\texttt{nodeState}_j \leftarrow RN_{j,m}$;
15: Send $\text{LINF}_{i,j,m}$ to CH_m for $MN_{i,m}$;
16: Send $\text{LACC}_{i,j,m}$ to $MN_{i,m}$;
17: **end if**

including member nodes and clusterhead, to all nodes in cluster C_1. Since the clusterhead CH_1 has the highest clustering metric, it remains as the clusterhead in the newly created cluster. Subsequently, it selects master and backup channels, and broadcasts `CHinfo`.

Next, the clusterhead CH_1 identifies common channels among remaining nodes in the cluster. It identifies that $MN_{4,1}$ and $MN_{5,1}$ are the remaining nodes having two channels (i.e., channels 4 and 7) in common. Since these channels fulfils the threshold $H_{C,min} = 2$, therefore the clusterhead CH_1 creates another cluster for them (i.e., cluster C_2 in Figure 7 after cluster splitting) and selects $MN_{5,1}$ as a clusterhead for cluster C_2 as it has the highest clustering metric. Subsequently, it sends clustering message `CHinfo` containing information of the new cluster C_2, including member nodes and clusterhead, to these remaining nodes of the cluster. Finally, at the end of cluster splitting, there are two clusters (e.g., C_1 and C_2), one is comprised of four nodes and the other is comprised of two nodes, as shown in Figure 7.

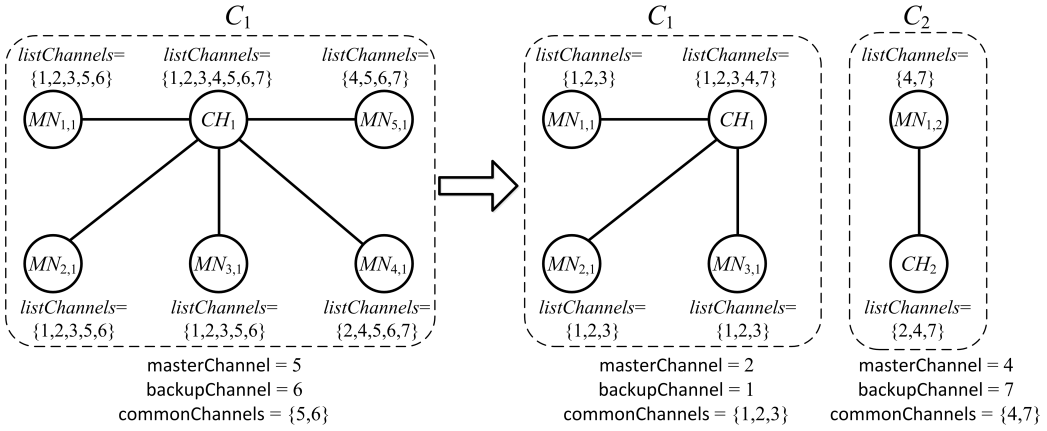

Figure 7: An example of cluster splitting from cluster C_1 to clusters C_1 and C_2.

5.6.2. Procedure

Algorithm 4 represents cluster splitting procedure for cluster C_j at clusterhead CH_j. A clusterhead has knowledge about the list of available channels of all nodes in its cluster. In part I of Algorithm 4, when clusterhead CH_j determines that cluster splitting is required (i.e., $n_{C,C_j} < H_{C,min}$), it counts the number of nodes $N_{N \in C_j,k}$ in each available channel $k \in listChannels_{C_j}$ and ranks these channels based on maximum node degree (i.e., $\gamma^t_{chan,k} > \gamma^t_{chan,l}$ if $N_{N \in C_j,k} > N_{N \in C_j,l} \mid k \in listChannels_{C_j}, l \in listChannels_{C_j} - k$). Then, it selects a certain number of most favourable channels as common channels of the new cluster C_k. The number of selected channels must satisfy the threshold for the minimum number of common channels in a cluster $H_{C,min}$ (i.e., $\texttt{commonChannels}_{C_k} \leftarrow N_{\gamma^t_{chan,k}} = 1, 2, ..., H_{C,min}$). The clusterhead then identifies a list of nodes ($\texttt{listNodes}_{C_k}$) which have these favourable channels available and forms a cluster comprised of these nodes (i.e., $C_k \leftarrow \texttt{listNodes}_{C_k}$).

Next, in part II of Algorithm 4, the clusterhead CH_j identifies the remaining nodes, which are yet to be clustered for new cluster C_l (i.e., $\texttt{listNodes}_{C_l} \leftarrow \texttt{listNodes}_{C_j} - \texttt{listNodes}_{C_k}$), and identifies a set of common channels among these nodes (i.e., $\texttt{commonChannels}_{C_l} \leftarrow \cap_{i=1}^{clusterSize_{C_l}} listChannels_{MN_{i,l}}$). The number of common channels must equal to the threshold for the minimum number of common channels in a cluster (i.e., $N_{\texttt{commonChannels}_{C_l}} = H_{C,min}$). The clusterhead forms another cluster comprised of these nodes $C_l \leftarrow \texttt{listNodes}_{C_l}$.

Finally, in part III of Algorithm 4, the clusterhead CH_j selects a node as clusterhead which has the highest clustering metric (i.e., the highest number of available channels among nodes in a cluster) for both newly created clusters C_k and C_l and broadcasts clustering message `CHinfo`. Subsequently, the new clusterheads CH_k and CH_l, elected by the old clusterhead CH_j, select their master and backup channels and broadcast `CHinfo`, which is not shown in the algorithm. Moreover, if the clusterhead CH_j is not elected as a clusterhead after cluster splitting, it relinquishes its role of clusterhead and becomes member node of its respective cluster (i.e., $\texttt{nodeState}_{CH_j} \leftarrow MN_{i,k}$ or $\texttt{nodeState}_{CH_j} \leftarrow MN_{i,l}$).

6. SMART: Routing

This section presents routing scheme of SMART which runs on clustered network. The cluster-based routing scheme is based on a RL-based routing model known as Q-routing, which was proposed by Boyan and Littman [30]. Q-routing is inspired by a RL approach [2] known as Q-learning. In this section, we propose a routing scheme that runs on clustered networks. The main objective of the routing scheme is to provide stable routes with higher OFF-state probabilities of channels along the routes in order to reduce SUs' transmission interruption due to PUs' activities, number of channel switches due to the re-appearance of PUs' activities and the occurrence of re-routing. This leads to enhanced SUs' network performance. Our approach is different from traditional Q-routing in a way that traditional Q-routing is based on end-to-end delay [30], while our approach is based on OFF-state probability of bottleneck channel along the route. To the best of our knowledge, the application of RL to cluster-based routing in CRNs is novel in its approach. In general, two routing schemes for CRNs that apply RL have been proposed [8, 9], and they are non-clustered in nature. In addition, while existing routing schemes select routes based on the number of available channels [8] and link-layer delay [9], SMART selects stable routes that have channels with higher OFF-state probability in order to minimize frequent disruptions and channel switches as a result of the re-appearance of PUs' activities during data communication. Most importantly, SMART is a novel joint channel selection and cluster-based routing scheme, while existing RL-based routing schemes assume that channel selection is readily available [8, 9].

In SMART, the Q-routing model is embedded in each clusterhead so that the clusterhead of a SU source node and the intermediate clusterheads can

Algorithm 4 Cluster splitting procedure at clusterhead CH_j

if $n_{C,C_j} < H_{C,min}$ **then**
  / Part I: Rank available channels based on maximum node degree */*
  Count $N_{N \in C_j,k} \mid k \in listChannels_{C_j}$;
  Rank channels $\gamma^t_{chan,k} > \gamma^t_{chan,l}$ if $N_{N \in C_j,k} > N_{N \in C_j,l} \mid k \in listChannels_{C_j}, l \in listChannels_{C_j} - k$;
  / Create a new cluster C_k */*
  $\texttt{commonChannels}_{C_k} \leftarrow N_{\gamma^t_{chan,k}} = 1, 2, ..., H_{C,min}$;
  Identify $\texttt{listNodes}_{C_k} | commonChannels_{C_k} \subseteq listChannels_{MN_{i,j}}, i \in listNodes_{C_j}$;
  Create cluster $C_k \leftarrow listNodes_{C_k}$;
  / Part II: Create second cluster C_l */*
  $\texttt{listNodes}_{C_l} \leftarrow \texttt{listNodes}_{C_j} - \texttt{listNodes}_{C_k}$;
  $\texttt{commonChannels}_{C_l} \leftarrow \cap_{i=1}^{clusterSize_{C_l}} listChannels_{MN_{i,l}} \mid N_{\texttt{commonChannels}_{C_l}} = H_{C,min}$;
  Create cluster $C_l \leftarrow listNodes_{C_l}$;
  / Part III: Select clusterheads for new clusters and broadcast* `CHinfo` **/*
  Select $MN_{i,k}$ as CH_k if $N_{listChannels_{MN_{i,k}}} > N_{listChannels_{MN_{j,k}}} \mid j \in \texttt{listNodes}_{C_k} - MN_{i,k}$;
  Select $MN_{i,l}$ as CH_l if $N_{listChannels_{MN_{i,l}}} > N_{listChannels_{MN_{j,l}}} \mid j \in \texttt{listNodes}_{C_l} - MN_{i,k}$;
  Broadcast clustering message `CHinfo`;
  if not ($CH_j = CH_k$ and $CH_j = CH_l$) **then**
    if $CH_j \in \texttt{nodeList}_{C_k}$ **then**
      $\texttt{nodeState}_{CH_j} \leftarrow MN_{i,k}$;
    else if $CH_j \in \texttt{nodeList}_{C_l}$ **then**
      $\texttt{nodeState}_{CH_j} \leftarrow MN_{i,l}$;
    end if
  end if
end if

make routing decisions (i.e., SU next-hop neighbor node selection) based on channel selection performed by clustering. The state s^t represents the SU destination node and the action a_i^t represents SU i's next-hop neighbor node that relays packets towards SU destination node s^t. At time t, a clusterhead i estimates the Q-value $Q_i^t(s^t, a_i^t)$ for each SU neighbor node, which indicates the OFF-state probability of the bottleneck channel along the route and updates its routing table of Q-values as shown in Table 5. The bottleneck channel is the channel having the least OFF-state probability for the next time slot along a route towards SU destination node s^t, connecting two clusters via a next-hop node a_i^t. Using the OFF-state probability of channels helps to reduce the effects of the dynamicity of channel availability in CRNs by selecting stable routes that have channels with higher OFF-state probability in order to maximize the utilization of white spaces (see C.1 in Section 1). Each column in Table 5 shows a SU neighbor node of SU source node i, while each row shows a destination node s^t. Each cell represents the Q-value of the next-hop neighbor node a_i^t selected by SU source node i to reach the destination node s^t. A clusterhead calculates Q-value for each SU neighbor node, while SU intermediate node calculates the OFF-state probability of the bottleneck channel from itself towards SU destination node s^t and forwards this probability to upstream node in RREP message. A clusterhead i makes route selection by selecting a next-hop SU neighbor node a_i^t with the maximum Q-value $Q_i^t(s^t, a_i^t)$ from the routing table of Q-values for destination node s^t. Upon sending RREQ to SU neighbor node a_i^t, a SU i receives OFF-state probability of the bottleneck channel for the route from SU neighbor node a_i^t to destination node s^t through RREP. A SU i updates its Q-value $Q_i^t(s^t, a_i^t)$ at time $t+1$ for destination node s^t via next-hop $a_i^t = j$ as follows:

$$Q_i^{t+1}(s^t, j) \leftarrow \big((1-\alpha) \times Q_i^t(s_i^t, j)\big) + \left[\alpha \times \min\left(r_i^{t+1}(j), \max_{k \in a_j^t} Q_j^t(s^t, k)\right)\right] \tag{3}$$

where $0 \leq \alpha \leq 1$ represents learning rate, $r_i^{t+1}(j)$ is the OFF-state probability of the operating channel between SU i and SU neighbor node j, $Q_j^t(s^t, k)$ is the OFF-state probability of the bottleneck channel along a route from SU node $k \in a_j^t$ (i.e., a SU next-hop neighbor node of SU j) to SU destination node s^t, and $\min\left(r_i^{t+1}(j), \max_{k \in a_j^t} Q_j^t(s^t, k)\right)$ represents OFF-state probabil-

Table 5: Routing table of Q-values at SU source node i

SU destination node s^t \ SU neighbor node a_i^t	1	2	3	m
1	—	$Q_i^t(1,2)$	$Q_i^t(1,3)$	$Q_i^t(1,m)$
2	$Q_i^t(2,1)$	—	$Q_i^t(2,3)$	$Q_i^t(2,m)$
3	$Q_i^t(3,1)$	$Q_i^t(3,2)$	—	$Q_i^t(3,m)$
n	$Q_i^t(3,1)$	$Q_i^t(n,2)$	$Q_i^t(n,3)$	$Q_i^t(n,m)$

ity of one of the bottleneck channels among $r_i^{t+1}(j)$ and $\max_{k \in a_j^t} Q_j^t(s^t, k)$. This means that either the link connecting SU i and SU j, or one of the links in the route established between SU j and SU destination node s^t is the bottleneck link.

Using this model, SUs learn about the routes on the fly. Due to different levels of PUs' activities, the routes may have different OFF-state probabilities of channels. Thus, the selected route will have higher OFF-state probabilities of channels which will make the route stable as compared to other available routes. The rest of this section presents an example of the proposed cluster-based routing using RL, and procedures for route discovery and selection.

6.1. Example

Figure 8 shows a cluster-based routing example in which SU source node $MN_{1,1}$ sends data packets to SU destination node BS. Clusterhead CH_1 initiates route discovery by broadcasting RREQ, with route record list $\text{RREQ}_{RREC,CH_1,\text{BS}} = CH_1$, on its `masterChannel`=1. When gateway nodes $GN_{1,1,2}$ and $GN_{2,1,3}$ receive RREQ message, they forward it to their respective neighboring clusterheads CH_2 and CH_3, respectively.

When clusterhead CH_2 receives RREQ message, it appends its address in route record list (i.e., $\text{RREQ}_{RREC,CH_1,\text{BS}} = [CH_1, CH_2]$) and forwards it to its neighboring clusterhead CH_4 via its gateway node $GN_{2,2,4}$. Similar process runs on clusterheads CH_4 and CH_3 for RREQ propagation. Finally, there are two routes received at SU destination node BS, specifically routes $CH_1 - CH_2 - CH_4 -$ BS and $CH_1 - CH_3 -$ BS. The routes are performed at cluster level, therefore gateway nodes are not included in these routes. Subsequently, BS generates RREP messages and sends them back towards the SU source node CH_1 using the reverse routes that RREQ messages traverse.

When clusterhead CH_4 receives RREP message from SU destination node BS, it updates its Q-value with the OFF-state probability of operating channel between its gateway node $GN_{2,4,\text{BS}}$ and destination node BS (i.e., $Q^t_{CH_4}$ (BS,$\varnothing$) = 0.4). The next-hop of cluster C_4 is the SU destination node BS, which is represented by $\varnothing$. Subsequently, it embeds this Q-value in RREP and forwards it to downstream clusterhead CH_2 via its gateway node $GN_{1,4,2}$. When clusterhead CH_2 receives RREP from clusterhead CH_4 through its gateway node $GN_{2,2,4}$, it compares OFF-state probability provided by clusterhead CH_4 with OFF-state probability of the operating channel between the pair of gateway nodes $GN_{2,2,4}$ and $GN_{1,4,2}$ for reaching clusterhead CH_4, and finds that this communication channel has lower OFF-state probability (i.e., $\varphi^t_4 = 0.3 < \varphi^t_5 = 0.4$), therefore it updates the Q-value (i.e., $Q^t_{CH_2}$ (BS,CH_4) = 0.3), embeds it in the RREP and forwards it to the upstream node (i.e., SU source node CH_1) via gateway node $GN_{1,2,1}$.

When SU source node CH_1 receives RREP from clusterhead CH_2, it updates its routing table of Q-values. Since this is the first attempt of route discovery, therefore, the Q-value has been initialized $Q^{t-1}_{CH_1}$ (BS,CH_2) = 0. The OFF-state probability of operating channel between SU source node CH_1 and clusterhead CH_2 is $\varphi^t_2 = 0.5$, therefore $r^t_{CH_1}(CH_2) = 0.5$. Hence, using Equation (3) with $\alpha = 0.5$, which is discussed in Section 6, the Q-value $Q^t_{CH_1}(\text{BS}, CH_2)$ is updated by $Q^t_{CH_1}(\text{BS}, CH_2) \leftarrow ((1 - 0.5) \times 0) + (0.5 \times \min(0.5, 0.3)) = 0.15$. Similar process runs on clusterhead CH_3 to process RREP, hence at SU source node CH_1, Q-value is updated as $Q^t_{CH_1}(\text{BS}, CH_3) \leftarrow ((1 - 0.5) \times 0) + (0.5 \times \min(0.1, 0.4)) = 0.05$.

Finally, routing table of SU source node CH_1 is comprised of two entries, specifically, $Q^t_{CH_1}(\text{BS}, CH_2) = 0.15$ and $Q^t_{CH_1}(\text{BS}, CH_3) = 0.05$. It selects CH_2 as its next-hop SU node because it provides the highest Q-value for the route leading to SU destination node BS. Note that the route $CH_1 - CH_2 - CH_4 - \text{BS}$ is selected, although it is a longer route compared to route $CH_1 - CH_3 - \text{BS}$ because it is more stable route with higher OFF-state probability at bottleneck channel along the route.

6.2. Procedure

In SMART, RREQ message is used to find a route from SU source node to SU destination node if the SU source node is not aware of any route or the existing route towards the SU destination node is expired. RREP message is used to inform the SU source node about a route towards a SU destination node and the OFF-state probability of the bottleneck channel along this

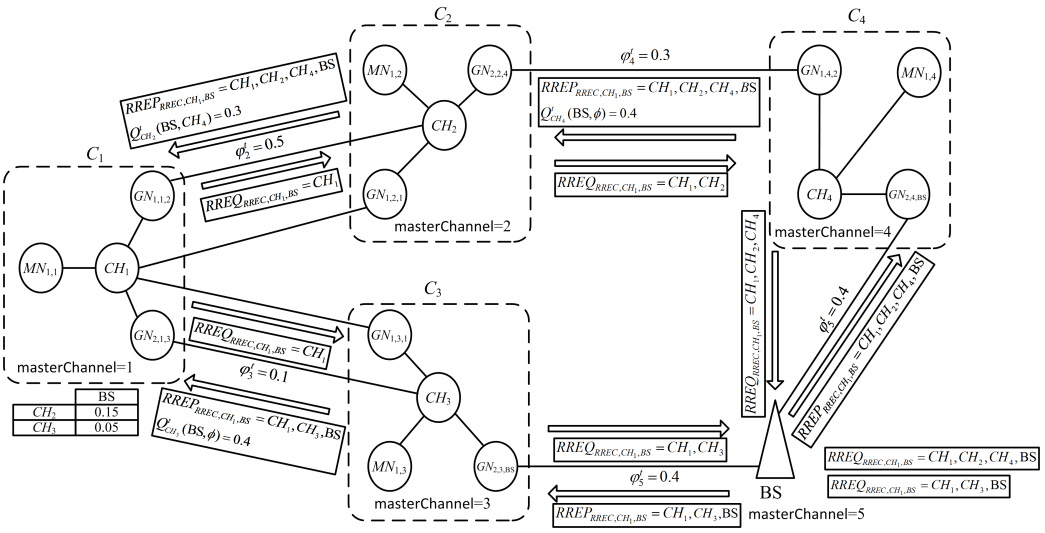

Figure 8: An example of routing from cluster C_1 to SU BS.

route. Since a SU member node sends its data to its clusterhead, which serves as a point of process for all member nodes in its cluster, we consider a SU clusterhead is the source node. Suppose, a clusterhead i does not know a route to destination node $s^t = l$ (i.e., $route_{i,l} = \varnothing$), so it triggers route discovery. It creates RREQ message and includes its own node ID in route record $\text{RREQ}_{RREC,i,l} \leftarrow i$. Then, it broadcasts the RREQ message on its master channel. When a gateway node $GN_{i,j,l}$ receives RREQ message from its clusterhead CH_i, it forwards RREQ message to its neighboring clusters.

In part I of Algorithm 5(a), when a SU intermediate clusterhead m receives RREQ message, it appends its own node ID m to the list of route record (i.e., $\text{RREQ}_{RREC,i,l} \leftarrow \text{RREQ}_{RREC,i,l} \cup m$). Subsequently, it rebroadcasts RREQ on its master channel if it is not the SU destination node l, which may be forwarded by gateway nodes to neighboring clusters. When RREQ is received by a SU destination node l, it generates RREP message using the route found in the RREQ message. In part II of Algorithm 5(a), when a SU intermediate node m receives RREP message from its next-hop neighbor node n, it calculates the OFF-state probability of the bottleneck channel from itself to SU destination node $s^t = l$ via next-hop neighbor node n (i.e., $Q_m^t(l, n)$), embeds it into RREP and forwards the RREP to the downstream node in the list of route record $\text{RREP}_{RREC,i,l}$. The RREP message follows the reverse route that the RREQ message traverses so that it reaches

the SU source node i. When SU source node i receives RREP, it updates its routing table of Q-values using Equation (3) for SU destination node s^t and performs route selection.

Algorithm 5(a) RREQ propagation and RREP processing at SU node m

```
/* Part I: RREQ propagation */
if receive RREQ and m ∉ RREQ_{RREC,i,l} then
    RREQ_{RREC,i,l} ← RREQ_{RREC,i,l} ∪ m;
    if m = l then
        /* l is the destination node */
        Create RREP;
        Send RREP using the reverse path in RREQ_{RREC,i,l};
    else
        Rebroadcast RREQ;
    end if
    /* Part II: RREP processing */
else if receive RREP from n then
    if m = i then
        /* i is the source node */
        Update Q_i^t(l, n) using Equation (3);
    else
        Calculate probability of bottleneck channel Q_m^t(l, n);
        Embed Q-value in RREP;
        Forward RREP;
    end if
end if
```

Once routes have been discovered and routing table of Q-values has been updated at SU source node i, the SU source node i selects a next-hop clusterhead with the highest Q-value from the routing table of Q-values for SU destination node s^t and starts data transmission along the selected route.

7. Performance Evaluation, Results and Discussion

In this section, we evaluate the performance of SMART through extensive simulations and present simulation setup and parameters, performance metrics, and performance evaluation.

7.1. Simulation Setup and Parameters

This section presents simulation setup and parameters. SMART is implemented in network simulator QualNet 6.1 [31]. QualNet 6.1, which is lack of support for CR functionality, is incorporated with CR functionality (see Section 4.1). SUs organize themselves into different clusters and each PU operates on a single channel. When PU starts transmission, its operating channel becomes unavailable to SUs in its communication range. The underlying physical layer model is IEEE 802.11b. The number of PUs ranges from 5 to 50, the number of SUs ranges from 10 to 100, and the total number of channels varies from 2 to 10. The default number of SUs is N=15, PUs is J=10, channels is K=5, threshold for the minimum number of common channels is $H_{C,min} = 2$ and cluster merging threshold is $H_{C,merge} = 4$. The transmission range of each PU and SU is 250m. In a cluster, whenever a master channel is occupied by PUs' activities, all member nodes and clusterhead in a cluster switch to a backup channel. In the network, PUs and SUs are randomly deployed in a square area of 1000m×1000m. The SU learning rate α is set to 0.5 in order to achieve a balance between recent and estimated values.

Each simulation run has a fixed SU source and SU destination nodes. Since the focus of our work is on the network layer, we assume a perfect channel sensing [32], scheduling and synchronization capabilities among SUs, as well as noiseless channels. For control information exchange, we assume there is an out-of-band common control channel which is free from PUs' activities all the times for this purpose. PU activity module (see Figure 3) generates exponentially distributed ON and OFF periods for each channel to represent activities in each channel (see Section 4.1.2.2) [25, 27]. We perform each simulation scenario for 100 times with different seeds, and each simulation runs for 550s. We adopt a constant bit rate (CBR) traffic in which a total of 500 packets are sent from SU source node to SU destination node and each packet is 512 bytes in size with an interval of 1s in between.

7.2. Performance Metrics

We compare the network performance achieved by SMART with clustered and non-clustered schemes. The clustered scheme, called SMART-NO-MNT (SMART no maintenance) works similarly with SMART but it does not perform cluster maintenance (i.e., cluster merging and cluster splitting). We choose SMART-NO-MNT to investigate the effects of cluster maintenance

Table 6: SMART simulation parameters and values.

Category	Parameter	Value
SU	Number of SUs	15
	Transmission range	250m
PU	Number of PUs	10
	Transmission range	250m
Channels	Number of total channels	5
Network	Area	1000m×1000m
	Placement of nodes	Random
	Active connections	1
	Traffic data type	UDP-CBR
	Packet size	512 bytes

on network performance achieved by SMART. The purpose of selecting SA-AODV is to compare SMART with non-clustered scheme designed for CR environment. The non-clustered scheme is a variant of the ad-hoc on-demand distance vector (AODV) routing protocol, known as SA-AODV (spectrum-aware AODV), which is developed for CR environment. SA-AODV is similar to AODV except that it is spectrum-aware and it operates on multi-channel environment. It selects an operating channel randomly from a list of available channels at each SU. Note that, SA-AODV has been extensively used in the literature for comparison such as [23]. SMART differs from SA-AODV in many perspectives. Firstly, in SA-AODV, a node broadcasts RREQ message to all neighboring nodes, which further broadcast it to their neighboring nodes, while SMART performs selective broadcasting in which clusterhead transmits RREQ message to its gateway nodes in a cluster only, and gateway nodes forward RREQ to their respective neighboring clusters. In this manner, SMART solves the problem associated with broadcasting using a single transceiver in CRNs (see C.3 in Section 1). Secondly, in SA-AODV, route decision is made by the destination node, while in SMART, route decision (i.e., next-hop clusterhead selection) is made by the clusterhead of SU source node based on Q-values. Thirdly, SA-AODV is a single-path routing protocol while SMART is a multipath routing protocol. Fourthly, SA-AODV is a non-cluster-based routing protocol while SMART is a cluster-based routing protocol.

The performance metrics for SMART are as follows:

- *SU-PU interference ratio* is the ratio of the total number of SUs' packets

interfered with PUs' activities to the total number of packets sent by a SU source node. Whenever a SU's packet interferes with a PU's packet, both PU's and SU's packets are lost, and so lower SU-PU interference ratio is favourable.

- *Route discovery frequency* is the number of route discovery (or RREQ) messages generated by a SU source node, and so lower route discovery frequency is favourable because it indicates that the routes have greater stability and lower routing overhead.

- *Throughput* is defined as the total number of bits received per second by a SU destination node. Higher throughput is favourable.

- *End-to-end delay* is calculated by dividing the total delay of all packets received at a SU destination node by total number of packets sent from a SU source node. Lower end-to-end delay is favourable.

7.3. Performance Evaluation

This section compares the performance of SMART with SMART-NO-MNT and SA-AODV under the varying effects of PUs' activities, number of channels and number of PUs.

7.3.1. Effects of PUs' activities

SUs exploit white spaces of licensed channels which may have different types of PUs' activities. This section investigates the effects of different types of PUs' activities (see Section 4.1.2.2) on the network performance of SMART, SMART-NO-MNT and SA-AODV.

Figure 9 shows the SU-PU interference ratio under different types of PUs' activities for the three schemes. SMART causes a significantly lower SU-PU interference as compared to SA-AODV for different types of PUs' activities. There are two main reasons. Firstly, SMART selects routes that have channels with higher OFF-state probability for the next time slot. Secondly, SMART applies RL which helps SUs to make right decisions on SU next-hop selection in routing. Since SA-AODV selects channels randomly, despite being spectrum-aware, the selected routes may have channels with lower OFF-state probability, and so there is a higher chance that PUs may re-appear on the selected channels in the next time slot, which causes higher interference to PUs. SMART-NO-MNT achieves lower SU-PU interference ratio as compared to SMART. This is because SMART-NO-MNT does not perform

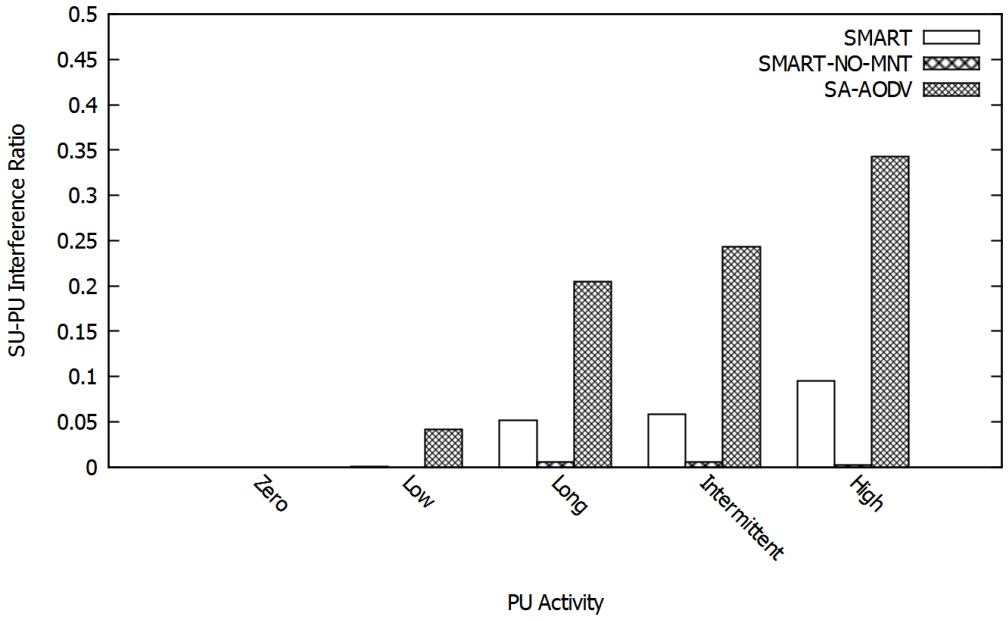

Figure 9: SU-PU interference ratio versus PUs' activities.

cluster maintenance; therefore, clusters may lack of a common channel available to all SUs in a cluster for intra-cluster communication. Hence, whenever a member node transmits packet to its clusterhead, the clusterhead may not find an appropriate gateway node to forward packet to neighboring clusters, and so it drops the packet. With lower number of transmissions, the SU-PU interference is naturally lower, which is reflected in the lower throughput (see Figure 11).

Figure 10 shows route discovery frequency under different types of PUs' activities for the three schemes. SMART achieves significantly lower route discovery frequency as compared to SMART-NO-MNT and SA-AODV. This is because SMART selects stable routes that have channels with higher OFF-state probability; therefore, there is less chance that routes get affected more frequently by PUs' activities. SMART-NO-MNT causes higher route discovery frequency as compared to SMART. This is because SMART-NO-MNT does not perform cluster maintenance mechanism; therefore, it lacks intra-cluster connectivity most of the times due to PUs' activities. Hence, whenever a SU source node has data to transmit, it is likely that it cannot find a route towards SU destination node due to varying PUs' activities, and so

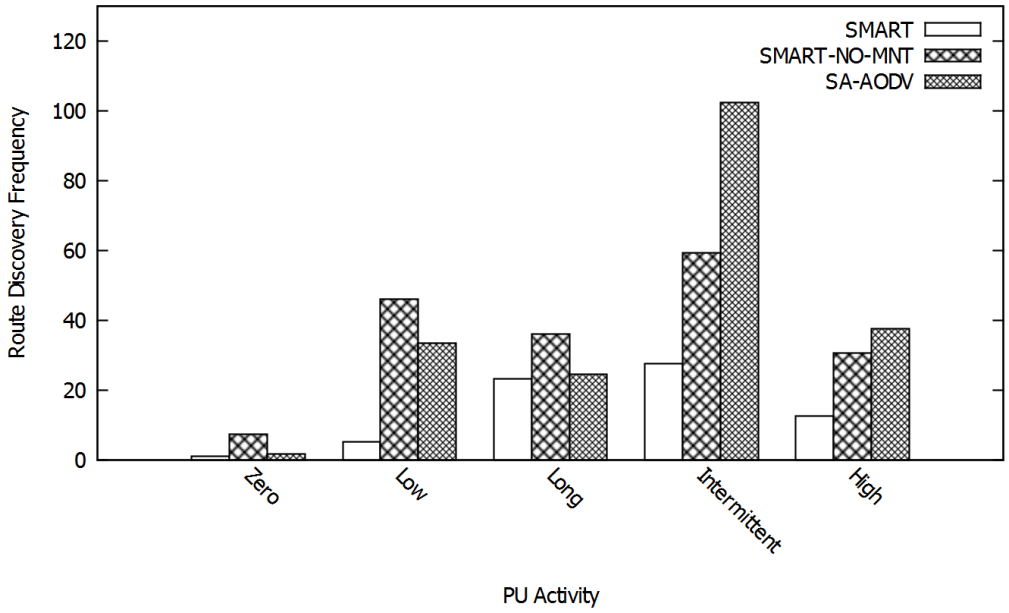

Figure 10: Route discovery frequency versus PUs' activities.

it initiates RREQ which causes higher route discovery frequency. SA-AODV achieves similar route discovery frequency as SMART-NO-MNT except at intermittent PU activity in which route discovery frequency by SA-AODV is significantly higher. There are two main reasons. Firstly, SA-AODV selects channels randomly, therefore, the selected routes may not contain channels with higher OFF-state probability, and so there is a higher chance that routes get affected more frequently by PUs' activities. Secondly, SA-AODV maintains only one route towards SU destination node. Since PUs constantly appear and disappear on very short durations in intermittent PU activity, the established routes in SA-AODV get affected more frequently by PUs' activities.

Figure 11 shows throughput under different types of PUs' activities for the three schemes. When PU activity level is zero (or no PUs' activities), SMART and SA-AODV achieve similar throughput. When PU activity level is low, SMART achieves higher throughput as compared to SA-AODV. This is because the effect of low PU activity level is not significant in SMART due to the selection of stable routes that have channels with higher OFF-state probability. SMART-NO-MNT achieves lower throughput for all types of PUs' ac-

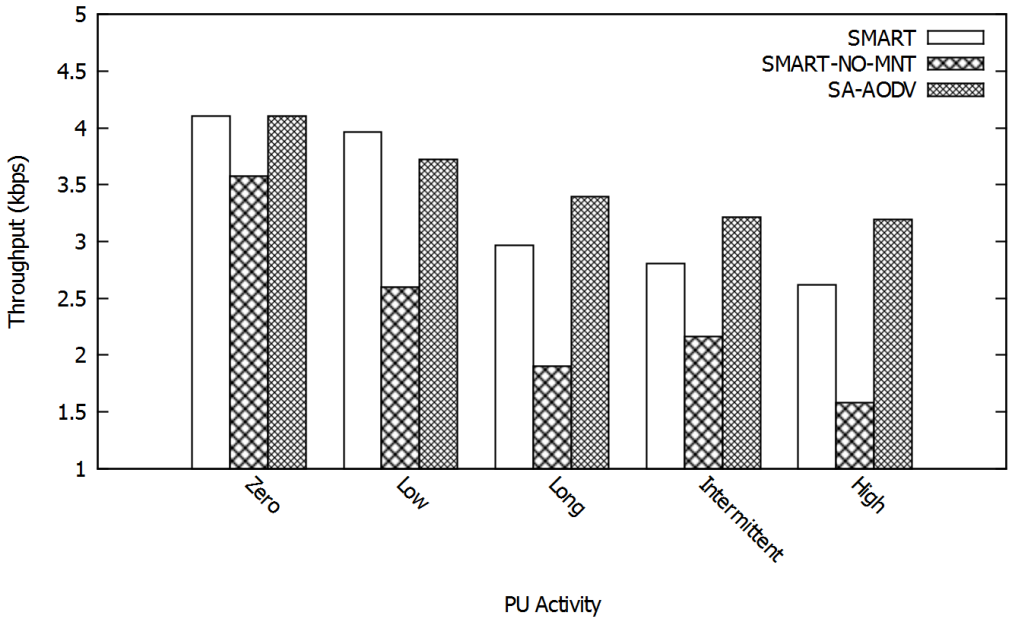

Figure 11: Throughput versus PUs' activities.

tivities even when there is no PU activity. This is because SMART-NO-MNT may not have a common channel in a cluster for intra-cluster communications most of the times which causes higher packet loss. When PUs' activities levels are long, intermittent and high, SMART achieves lower throughput as compared to SA-AODV because SMART incurs clustering delay and saves PUs from interference, while SA-AODV achieves higher throughput at the expense of higher SU-PU interference (see Figure 9).

Figure 12 shows end-to-end delay under different types of PUs' activities for the three schemes. In this figure, when PU activity level is zero (or no PUs' activities), all schemes achieve significantly lower end-to-end delay because no route is affected by PUs' activities and all schemes can transmit their packets without disruptions. When PU activity level is low, SMART-NO-MNT causes significantly higher end-to-end delay. This is because, due to the low level of PU activity, SMART-NO-MNT has intra-cluster connectivity most of the times, however, it need to perform more packet retransmissions, and so it causes higher end-to-end delay. For all types of PUs' activities, SA-AODV achieves significantly lower end-to-end delay. This is because SA-AODV is a non-clustered scheme, therefore, it does not incur higher delay

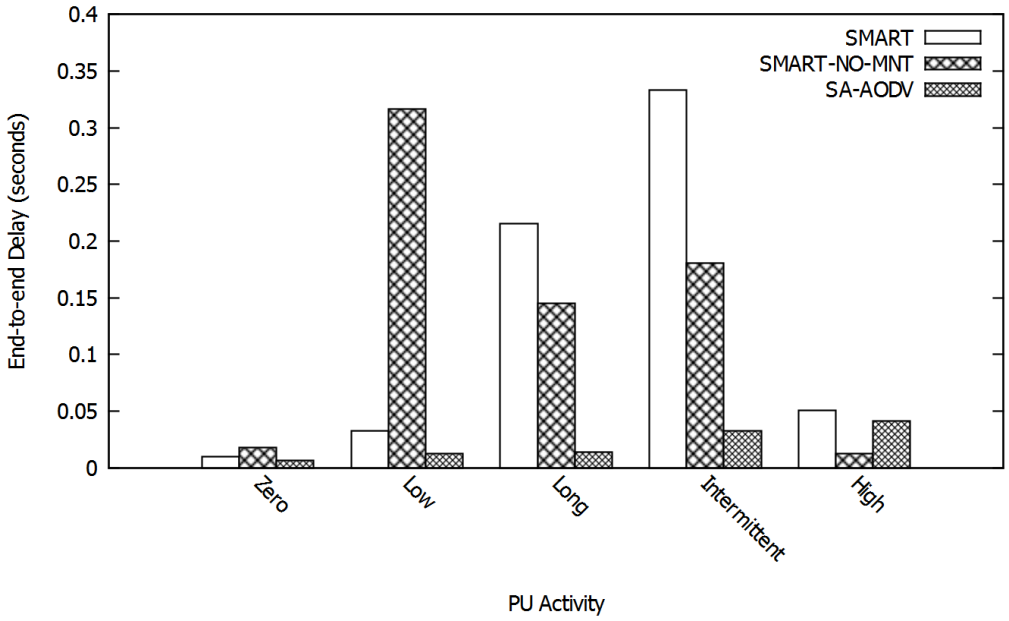

Figure 12: End-to-end delay versus PUs' activities.

caused by clustering, in which a member node first transmits packet to its clusterhead, subsequently the clusterhead forwards packet to an appropriate gateway node, and gateway node further forwards the packet towards neighboring cluster. Therefore, as expected, clustering schemes incur higher end-to-end delay as compared to non-clustered schemes. SMART achieves higher end-to-end delay as compared to SMART-NO-MNT when PUs' activities levels are long, intermittent and high. This is because for these types of PUs' activities, SMART-NO-MNT drops higher number of packets due to the lack of either intra-cluster connectivity or route towards SU destination node. Therefore, with lower number of transmission opportunities, the end-to-end delay is naturally lower in SMART-NO-MNT, while SMART adapts clusters based on varying PUs' activities, so it causes higher end-to-end delay due to frequent cluster maintenance.

In summary, this section investigates the performance of SMART in the presence of different types of PUs' activities. SMART outperforms SMART-NO-MNT and SA-AODV for varying PUs' activities and reduces routing overhead in terms of route discovery frequency by selecting stable routes that have channels with higher OFF-state probability. SMART achieves signifi-

Table 7: Threshold values for varying number of channels.

Threshold	Number of channels				
	2	4	6	8	10
$H_{C,min}$	1	2	4	6	8
$H_{C,merge}$	2	3	5	7	9

cantly lower interference to PUs without significant degradation of throughput and end-to-end delay. However, when PUs' activity levels are long and intermittent, SMART incurs higher end-to-end delay, which is our future work.

7.3.2. Effects of number of channels

SUs explore the availability of multiple channels in the network and exploit them. This section investigates the effects of varying number of channels on network performance of SMART, SMART-NO-MNT and SA-AODV. For a varying number of channels, each channel contains an equal number of PUs, and there are two thresholds, namely the minimum number of common channels in a cluster ($H_{C,min}$), which is the threshold for cluster formation and cluster splitting, and the minimum number of common channels in a cluster required to initiate cluster merging ($H_{C,merge}$). These thresholds values are presented in Table 7. Note that, for a certain number of channels, the value $H_{C,min}$ is one less than $H_{C,merge}$ to avoid frequent cluster merging and splitting because if $H_{C,min} = H_{C,merge}$, cluster merging and splitting may occur even when a PU re-appears in a single common channel in a cluster. Furthermore, the values of threshold $H_{C,min}$ and $H_{C,merge}$ increase with the increasing number of channels in a network so that SUs can exploit the availability of multiple channels in the network. This enhances cluster stability by maintaining higher number of common channels in a cluster (see C.2 in Section 1).

Figure 13 shows SU-PU interference ratio by varying the number of channels in the network for the three schemes. When there are 2 channels in the network, SMART achieves lower SU-PU interference ratio as compared to other number of channels because of lower number of transmission, specifically, there is higher chance that a SU does not find an available channel for transmission. When there are 4, 6, 8 and 10 channels in the network, a SU has sufficient number of channels for transmissions, and so SMART causes slightly higher SU-PU interference as compared to the case of 2 channels

in the network due to higher number of transmissions. However, SMART achieves lower SU-PU interference with the increasing number of channels in the network. This is because SMART exploits the availability of multiple channels in the network, and so it achieves lower SU-PU interference ratio with higher number of channels. In SMART-NO-MNT, when there are 2 channels in the network, it causes significantly higher SU-PU interference ratio as those with higher number of channels in the network. This is because, since SMART-NO-MNT does not perform cluster maintenance, and with only 2 channels in the network, there is a higher chance that neighboring clusters are also operating on similar channels. Hence, there are intra-cluster and inter-cluster connectivities, and SU source node can transmit packets towards SU destination node. However, with only 2 channels, there is higher SU-PU interference. Note that, SMART achieves significantly lower SU-PU interference when there are 2 channels in the network due to lower number of transmissions because SMART performs cluster maintenance frequently to overcome the dynamicity of channels availability.

When there are 6, 8 and 10 channels in the network, SMART-NO-MNT does not cause SU-PU interference. There are two main reasons. Firstly, when there is higher number of channels, there is a higher chance that neighboring clusters are operating on different channels, and so SMART-NO-MNT lacks inter-cluster connectivity most of the times. Therefore, there is lower number of transmissions causing higher number of packet loss and lower SU-PU interference ratio. Secondly, when there is higher number of channels, SMART-NO-MNT exploits the availability of multiple channels, and so there is less chance of SU-PU interference. SA-AODV achieves lower SU-PU interference ratio with the increasing number of channels in the network. This is because SA-AODV also operates on multiple channels; therefore, with higher number of channels, there is less chance of SU-PU interference. Overall, SA-AODV achieves significantly higher SU-PU interference for all number of channels as compared to SMART and SMART-NO-MNT due to random channel selection, which may choose non-stable routes that have channels with lower OFF-state probability.

Figure 14 shows route discovery frequency by varying the number of channels in the network for the three schemes. When there are 2 channels in the network, SMART achieves significantly lower route discovery frequency. This is because SMART selects stable routes that have channels with higher OFF-state probability and performs cluster maintenance in order to reduce dynamic effects of the network. The route discovery frequency in SMART is

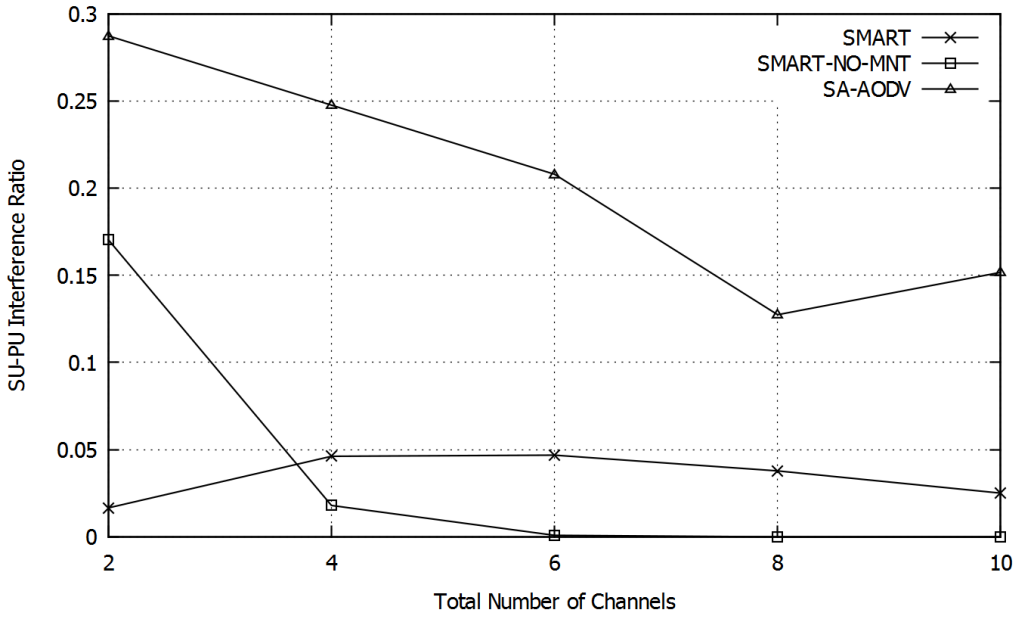

Figure 13: SU-PU interference versus varying number of channels.

slightly lower when there are 2 channels as compared to 4 channels. This is because, when there are lower number of channels in the network, SMART performs cluster maintenance frequently, and so less RREQs are initiated. SMART-NO-MNT causes significantly higher route discovery frequency when there are 2 channels in the network because it does not perform cluster maintenance, higher number of RREQs are initiated. SMART and SMART-NO-MNT achieve lower and similar route discovery frequency for higher number of channels. This is because both schemes exploit availability of multiple channels and select stable routes that have channels with higher OFF-state probability. The reason of similar level of route discovery frequency is that higher number of channels increases the number of common channels in a cluster which maximizes stability naturally (see C.2 in Section 1) and reduces the need of frequent cluster maintenance, and so SMART-NO-MNT performs well without cluster maintenance. SA-AODV also achieves lower route discovery frequency for higher number of channels. This is because it exploits the availability of multiple channels. However, it causes higher route discovery frequency as compared to SMART and SMART-NO-MNT because it does not select stable routes.

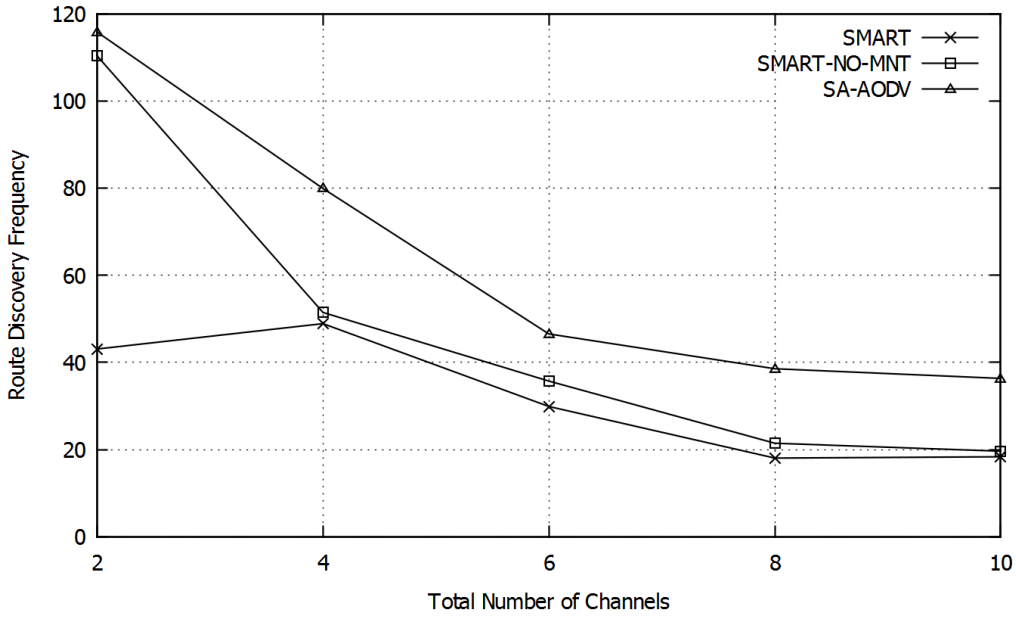

Figure 14: Route discovery frequency versus varying number of channels.

Figure 15 shows throughput by varying the number of channels in the network for the three schemes. When there are 2 channels in the network, SMART achieves significantly lower throughput as compared to networks with higher number of channels, and other schemes, i.e., SMART-NO-MNT and SA-AODV. This is because, when there are 2 channels in the network, a cluster can have at most 2 common channels, and so there is a higher chance that a cluster gets affected more frequently by PUs' activities as compared to higher number of common channels. Therefore, due to lower number of channels, SMART performs cluster maintenance frequently to overcome the dynamic effects of network, which causes higher number of packet loss, and so SMART achieves lower throughput. SMART-NO-MNT and SA-AODV achieve higher throughput when there are 2 channels in the network as compared to SMART. This is because, since they do not perform cluster maintenance, therefore, there is no packet loss caused by frequent cluster maintenance, and so they achieve higher throughput as compared to SMART. When the number of channels increases from 4 to 10, SMART achieves significantly higher throughput as compared to a network with 2 channels because it performs cluster maintenance and adapts to higher num-

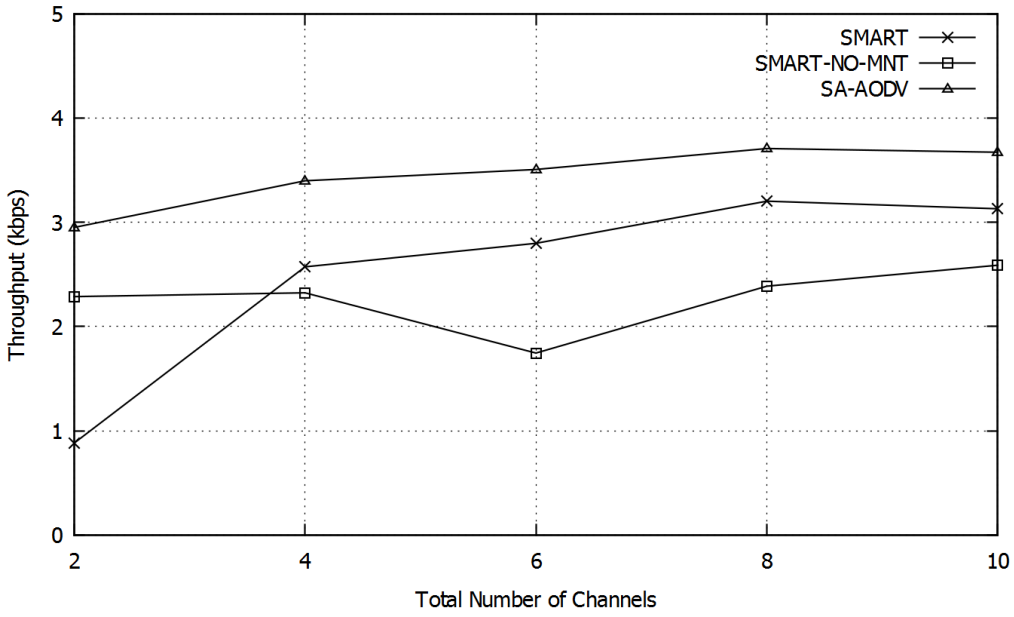

Figure 15: Throughput versus varying number of channels.

ber of available channels. SA-AODV achieves higher throughput with the increasing number of channels in the network because there is less chance of SU-PU interference. SMART-NO-MNT achieves similar throughput for varying number of channels due to the lack of cluster maintenance, and so it does not take the advantage of higher number of channels in the network by adapting cluster size based on the availability of multiple channels in the network. Moreover, the initially formed clusters may be suboptimal which causes lower throughput, as compared to SMART.

Figure 16 shows end-to-end delay by varying the number of channels in the network for the three schemes. When there are 2 channels in the network, SMART achieves significantly lower end-to-end delay as compared to SMART-NO-MNT and SA-AODV. This is because SMART cannot find an available channel for transmissions most of the times due to lower number of channels, therefore, it drops higher number of packets. The lower number of transmissions causes lower end-to-end delay. SMART-NO-MNT and SA-AODV may also not find an available channel for transmissions most of the times, but since they do not perform cluster maintenance, they initiate higher number of RREQs to discover new routes, which causes higher

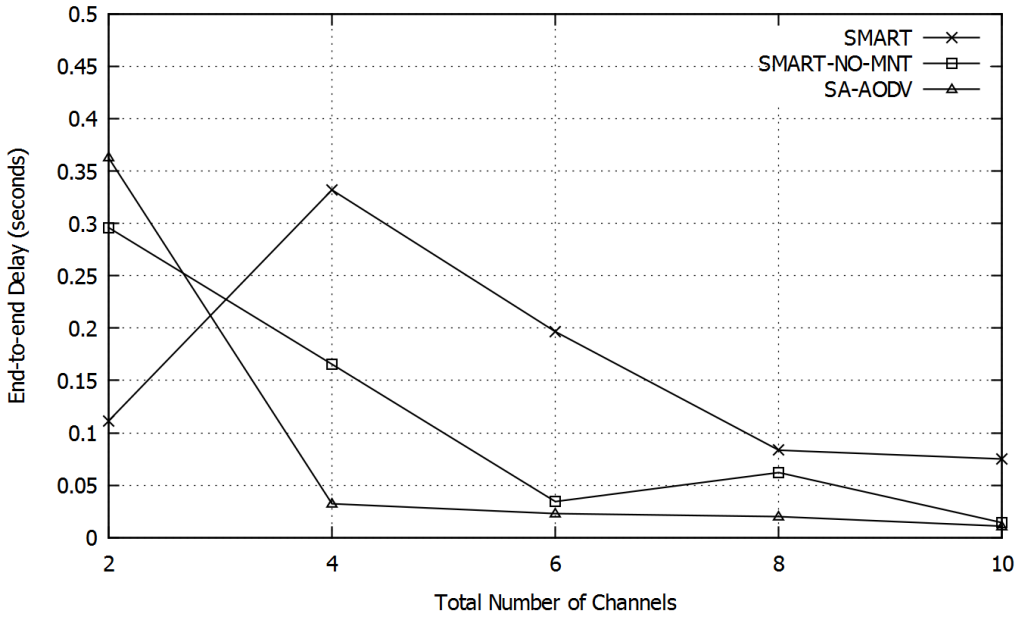

Figure 16: End-to-end delay versus varying number of channels.

end-to-end delay. When the number of channels increases from 4 to 10, all three schemes cause lower end-to-end delay. However, SA-AODV achieves significantly lower end-to-end delay for the increasing number of channels as compared to SMART and SMART-NO-MNT. This is because SA-AODV is a non-clustered scheme; therefore, it does not incur delay caused by clustering mechanisms. SMART causes higher end-to-end delay as compared to SMART-NO-MNT. This is because SMART-NO-MNT does not perform cluster maintenance, therefore, it drops higher number of packets due to the lack of either intra-cluster connectivity or a route towards SU destination node. On the other hand, SMART performs cluster maintenance to reduce the dynamic effects of network, therefore, it is able to find route towards SU destination node most of the times and transmit higher number of packets. Therefore, higher number of transmissions and cluster maintenance cause higher end-to-end delay.

In summary, this section investigates the performance of SMART for varying number of channels in the network. SMART outperforms SMART-NO-MNT and SA-AODV for varying number of channels and achieves lower SU-PU interference ratio and lower route discovery frequency without signif-

icant degradation of throughput and end-to-end delay. However, when there are 2 channels in the network, the performance enhancement brought about by SMART is not substantial due to lower number of channels and frequent cluster maintenance. Therefore, cluster maintenance is not encouraged for lower number of channels.

7.3.3. Effects of number of PUs

SUs explore and exploit the channels of PUs and a number of PUs may operate on similar channels at different geographic locations. This section investigates the effects of the number of PUs on the network performance of SMART, SMART-NO-MNT and SA-AODV. We consider there are a total of 5 channels in the network, and so PUs are equally distributed on each channel for varying number of PUs. For example, when there are a total 5 PUs in the network, each channel contains 1 PU. Similarly, when there are a total of 10 PUs in the network, each channel contains 2 PUs.

Figure 17 shows SU-PU interference ratio by varying the number of PUs in the network for the three schemes. SMART and SMART-NO-MNT achieve lower, similar and almost constant SU-PU interference ratio for varying numbers of PUs. This is because, SMART and SMART-NO-MNT select stable routes that have channels with higher OFF-state probability, so there is less chance of interference to PUs, and varying number of PUs has no significant effect on SU-PU interference. However, SA-AODV causes significantly higher SU-PU interference ratio as compared to SMART and SMART-NO-MNT. This is because SA-AODV selects channels randomly; therefore, there is a higher chance that selected routes may have channels with lower OFF-state probability.

Figure 18 shows route discovery frequency by varying the number of PUs in the network for the three schemes. SMART achieves significantly lower and almost constant route discovery frequency for varying number of PUs. This is because SMART selects stable routes that have channels with higher OFF-state probability; therefore, there is less chance that routes get affected more frequently due to PUs' activities, which causes lower route discovery frequency as compared to SA-AODV. SMART-NO-MNT causes higher route discovery frequency with the increasing number of PUs. This is because SMART-NO-MNT does not perform cluster maintenance; therefore, there is a higher chance that clusters are lack of inter-cluster connectivity. More specifically, when there are more PUs in the network, each cluster may operate in the presence of higher number of PUs, and so the master channel of

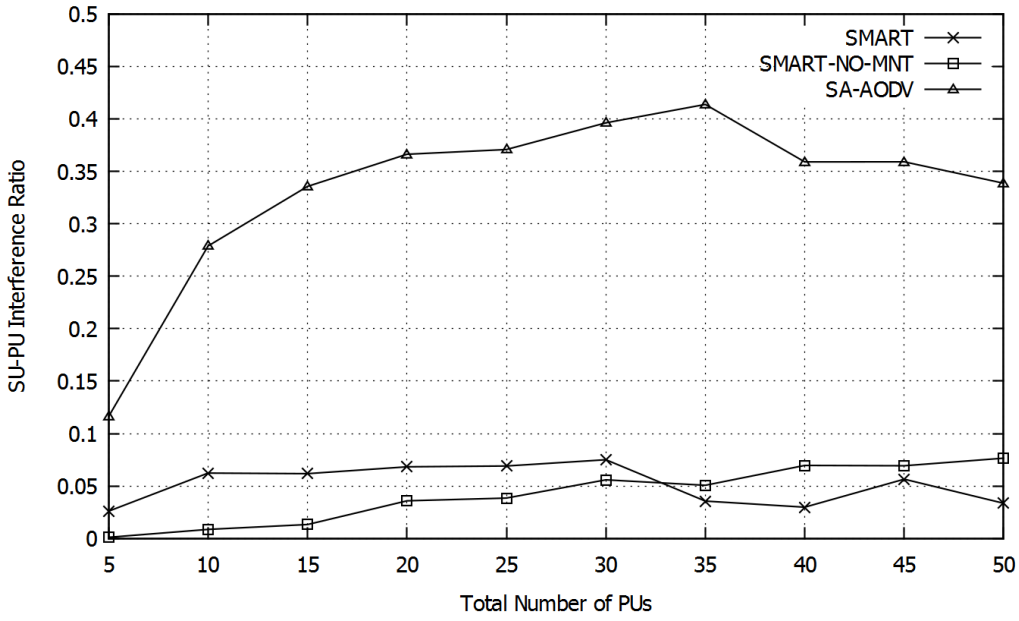

Figure 17: SU-PU interference ratio versus varying number of PUs.

a cluster changes more frequently due to higher dynamicity in the network caused by PUs' activities. Hence, there is higher chance that member nodes in a cluster do not have the master channel of a neighboring cluster for inter-cluster connectivity, which causes the lack of gateway nodes. Therefore, a SU source node initiates higher number of RREQs with the increasing number of PUs. SA-AODV causes significantly higher route discovery frequency with the increasing number of PUs. There are two main reasons. Firstly, SA-AODV does not select stable routes that have channels with higher OFF-state probability, and so there is higher chance that routes get affected more frequently due to PUs' activities. Secondly, it maintains a single route towards a SU destination node, and so it initiates RREQ every time a route gets affected due to PUs' activities.

Figure 19 shows throughput by varying the number of PUs in the network for the three schemes. SMART and SA-AODV achieve lower throughput with the increasing number of PUs. This is because, with the increasing number of PUs, there is a higher chance that SMART cannot find a route due to higher level of PUs' activities, and so higher number of packets are dropped. The reason of lower throughput achieved by SA-AODV with the increasing

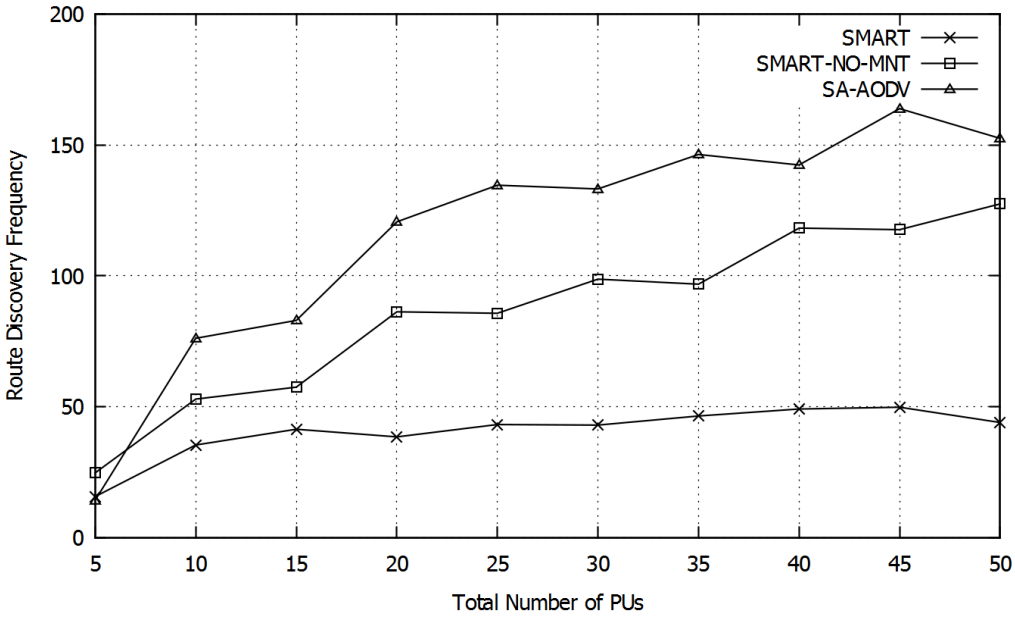

Figure 18: Route discovery frequency versus varying number of PUs.

number of PUs is that, SA-AODV causes higher SU-PU interference with the increasing number of PUs in the network, therefore, packet loss in SA-AODV is caused by interference with PUs. SMART-NO-MNT achieves nearly equal throughput for varying number of PUs. This is because, since SMART-NO-MNT does not perform cluster maintenance and a cluster is comprised of physically close SUs which may observe similar PUs' behaviour. Therefore, the increasing number of PUs does not have significant effect on intra-cluster connectivity due to the availability of backup and multiple common channels in a cluster.

Figure 20 shows end-to-end delay by varying the number of PUs in the network for the three schemes. SMART achieves higher end-to-end delay as compared to SMART-NO-MNT and SA-AODV. This is because SMART is a clustered scheme, therefore it causes higher end-to-end delay due to cluster maintenance, as the need of cluster maintenance increases with higher number of PUs. SMART-NO-MNT achieves slightly lower end-to-end delay as compared to SMART. This is because SMART-NO-MNT does not perform cluster maintenance, and so, it does not incurs such delay. SA-AODV achieves lower end-to-end delay. However, the effect of increasing number

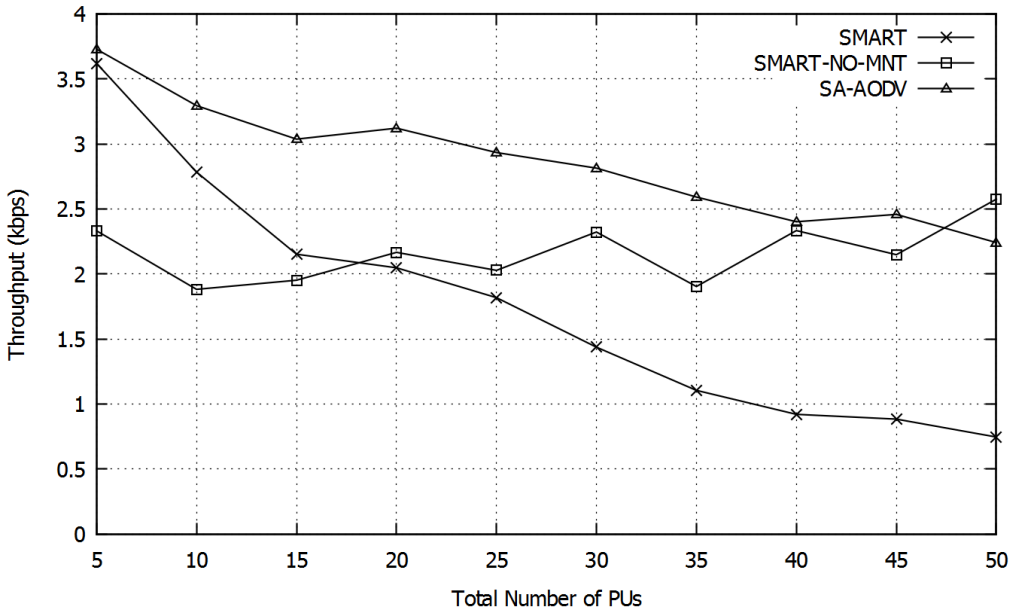

Figure 19: Throughput versus varying number of PUs.

of PUs on end-to-end delay is significantly higher as compared to SMART and SMART-NO-MNT, and so SA-AODV causes similar end-to-end delay as SMART when the number of PUs equal 50. This is because SA-AODV is a non-clustered scheme; therefore it achieves lower end-to-end delay when there are lower number of PUs in the network. However, when there are higher number of PUs in the network, there is a higher chance of route breakage, and so SA-AODV initiates higher number of RREQs which causes significantly higher end-to-end delay.

In summary, this section investigates the performance of SMART for varying number of PUs in the network in comparison with SMART-NO-MNT and SA-AODV. SMART achieves significantly lower SU-PU interference ratio and lower route discovery frequency. However, SMART causes slightly lower throughput and higher end-to-end delay. SMART (with cluster maintenance) is preferred when the number of PUs is low; while SMART-NO-MNT (without cluster maintenance) is preferred when the number of PUs is high, a scenario which is not preferable for CR operation.

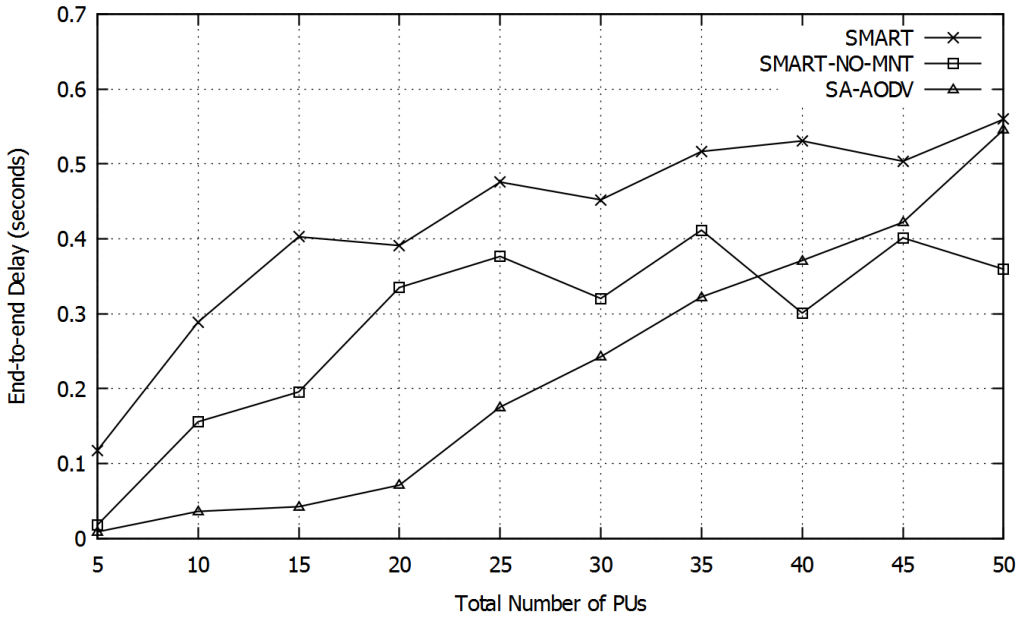

Figure 20: End-to-end delay versus varying number of PUs.

8. Conclusion and Future Work

This paper proposes a spectrum-aware cluster-based routing scheme called SMART for cognitive radio networks (CRNs). SMART enables secondary users (SUs) to form clusters in a CRN and enables each SU source node to search for a route to its destination node in a clustered network. SMART applies an artificial intelligence approach called reinforcement learning (RL) to help SUs to make right decisions on routing, particularly the selection of next-hop node based on the OFF-state probability of channel availability along the route by observing and learning the operating environment. SMART adapts cluster size by performing cluster maintenance, comprised of cluster merging and cluster splitting, to reduce the dynamic effects of the network (i.e., PUs' activities). The performance of SMART is compared with clustered scheme without cluster maintenance (SMART-NO-MNT) and non-clustered scheme (spectrum-aware AODV or SA-AODV). Simulation results show that SMART selects routes with higher stability, as well as reduces SUs' interference to primary users (PUs) and routing overhead in terms of route discovery frequency. These are achieved without significant degradation of throughput and end-to-end delay performances.

In future, we aim to investigate important areas of SMART. Firstly, cluster merging and cluster splitting can be made traffic-aware. Hence, cluster merging occurs if a cluster has enough capacity to handle traffics from its own and neighboring clusters, and cluster splitting occurs if a cluster does not have enough capacity to handle traffics from neighboring clusters. Secondly, best-fit channel selection (BFC) scheme [22] can be incorporated in order to select operating channel which will remain idle for the duration of a traffic, rather than the longest possible duration. Thirdly, channel capacity metric can be improved by considering exponential weighted moving average (EWMA) [33], which considers the most recent values in its estimation. Fourthly, SMART is designed for single-radio interface, so it can be extended to multi-radio interfaces. Fifthly, SMART can be extended to fulfil requirement on the minimum number of nodes in a cluster in order to enhance network scalability. Finally, stay-and-wait policy for channel selection can be incorporated so that SUs can stay in the current operating channel and wait for the PUs' activities to cease to exist.

Acknowledgement

This work was supported by the Malaysian Ministry of Science, Technology and Innovation (MOSTI) under Science Fund 01-02-16-SF0027. Kok-Lim Alvin Yau was also funded under Fundamental Research Grant Scheme (FRGS) FRGS/1/2014/ICT03/SYUC/02/2 and Small Grant Scheme (Sunway-Lancaster) SGSSL-FST-CSNS-0114-05.

References

[1] I. F. Akyildiz, W.-Y. Lee, K. R. Chowdhury, Crahns: Cognitive radio ad hoc networks, Ad Hoc Networks 7 (5) (2009) 810–836.

[2] R. S. Sutton, A. G. Barto, Introduction to reinforcement learning, MIT Press, 1998.

[3] M.-R. Kim, S.-J. Yoo, Distributed coordination protocol for common control channel selection in multichannel ad-hoc cognitive radio networks, in: IEEE WIMOB, 2009, pp. 227–232.

[4] A. Ephremides, J. E. Wieselthier, D. J. Baker, A design concept for reliable mobile radio networks with frequency hopping signaling, Proceedings of the IEEE 75 (1) (1987) 56–73.

[5] A. Jeng, R.-H. Jan, The r-neighborhood graph: An adjustable structure for topology control in wireless ad hoc networks, IEEE Transactions on Parallel and Distributed Systems 18 (4) (2007) 536–549.

[6] J. Wei, X. Zhang, Energy-efficient distributed spectrum sensing for wireless cognitive radio networks, in: IEEE INFOCOM Workshops, 2010, pp. 1–6.

[7] Y. Chen, A. Liestman, J. Liu, Clustering algorithms for ad hoc wireless networks, Ad Hoc and Sensor Networks 28.

[8] B. Xia, M. H. Wahab, Y. Yang, Z. Fan, M. Sooriyabandara, Reinforcement learning based spectrum-aware routing in multi-hop cognitive radio networks, in: CROWNCOM, IEEE, 2009, pp. 1–5.

[9] H. A. Al-Rawi, K.-L. A. Yau, Route selection for minimizing interference to primary users in cognitive radio networks: A reinforcement learning approach, in: IEEE Symposium on CIComms, IEEE, 2013, pp. 24–30.

[10] J. Y. Yu, P. H. J. Chong, A survey of clustering schemes for mobile ad hoc networks., IEEE Communications Surveys & Tutorials 7 (1-4) (2005) 32–48.

[11] R. G. Michael, S. J. David, Computers and intractability: a guide to the theory of np-completeness, WH Freeman & Co., San Francisco 18 (1979) 41.

[12] C. Peng, H. Zheng, B. Y. Zhao, Utilization and fairness in spectrum assignment for opportunistic spectrum access, Mobile Networks and Applications 11 (4) (2006) 555–576.

[13] H. Zhang, Z. Zhang, H. Dai, R. Yin, X. Chen, Distributed spectrum-aware clustering in cognitive radio sensor networks, in: IEEE GLOBECOM, 2011, pp. 1–6.

[14] X.-L. Huang, G. Wang, F. Hu, S. Kumar, Stability-capacity-adaptive routing for high-mobility multihop cognitive radio networks, IEEE Transactions on Vehicular Technology 60 (6) (2011) 2714–2729.

[15] W. Zhang, C. K. Yeo, Cluster-based adaptive multispectrum sensing and access in cognitive radio networks, Wireless Communications and Mobile Computing 15 (1) (2015) 100–114.

[16] M. Bradonjic, L. Lazos, Graph-based criteria for spectrum-aware clustering in cognitive radio networks, Ad Hoc Networks 10 (1) (2012) 75–94.

[17] O. S. Badarneh, H. B. Salameh, Probabilistic quality-aware routing in cognitive radio networks under dynamically varying spectrum opportunities, Computers & Electrical Engineering 38 (6) (2012) 1731–1744.

[18] A. C. Talay, D. T. Altilar, Self adaptive routing for dynamic spectrum access in cognitive radio networks, Journal of Network and Computer Applications 36 (4) (2013) 1140–1151.

[19] A. Bourdena, C. X. Mavromoustakis, G. Kormentzas, E. Pallis, G. Mastorakis, M. B. Yassein, A resource intensive traffic-aware scheme using energy-aware routing in cognitive radio networks, Future Generation Computer Systems 39 (2014) 16–28.

[20] A. C. Talay, D. T. Altilar, United nodes: cluster-based routing protocol for mobile cognitive radio networks, IET communications 5 (15) (2011) 2097–2105.

[21] A. S. Cacciapuoti, M. Caleffi, L. Paura, Reactive routing for mobile cognitive radio ad hoc networks, Ad Hoc Networks 10 (5) (2012) 803–815.

[22] S. Bayhan, F. Alagöz, A markovian approach for best-fit channel selection in cognitive radio networks, Ad Hoc Networks 12 (2014) 165–177.

[23] W. Kim, M. Gerla, S. Y. Oh, K. Lee, A. Kassler, Coroute: a new cognitive anypath vehicular routing protocol, Wireless Communications and Mobile Computing 11 (12) (2011) 1588–1602.

[24] A. W. Min, K. G. Shin, Exploiting multi-channel diversity in spectrum-agile networks, in: IEEE INFOCOM, 2008.

[25] M. H. Rehmani, A. C. Viana, H. Khalife, S. Fdida, Surf: A distributed channel selection strategy for data dissemination in multi-hop cognitive radio networks, Computer Communications 36 (10) (2013) 1172–1185.

[26] S. Bayhan, F. Alagöz, Distributed channel selection in crahns: A non-selfish scheme for mitigating spectrum fragmentation, Ad Hoc Networks 10 (5) (2012) 774–788.

[27] H. Kim, K. G. Shin, Efficient discovery of spectrum opportunities with mac-layer sensing in cognitive radio networks, IEEE Transactions on Mobile Computing 7 (5) (2008) 533–545.

[28] E. M. Belding-Royer, Multi-level hierarchies for scalable ad hoc routing, Wireless Networks 9 (5) (2003) 461–478.

[29] T. Chen, H. Zhang, G. M. Maggio, I. Chlamtac, Cogmesh: a cluster-based cognitive radio network, in: IEEE DySPAN, 2007, pp. 168–178.

[30] J. A. Boyan, M. L. Littman, Packet routing in dynamically changing networks: A reinforcement learning approach, in: Advances in neural information processing systems, 1994, pp. 671–671.

[31] Qualnet communications simulation platform, accessed: January 2015. URL http://web.scalable-networks.com/content/qualnet

[32] K. R. Chowdhury, I. F. Akyildiz, CRP: A routing protocol for cognitive radio ad hoc networks, IEEE Journal on Selected Areas in Communications 29 (4) (2011) 794–804.

[33] M. Hoyhtya, S. Pollin, A. Mammela, Classification-based predictive channel selection for cognitive radios, in: IEEE ICC, 2010, pp. 1–6.