Demand based State Aware Channel Reconfiguration Algorithm for Multi-Channel Multi-Radio Wireless Mesh Networks

A Antony Franklin
Elec. and Telecom. Res. Institute (ETRI), South Korea

Athula Balachandran
School of Computer Science CMU, USA

C. Siva Ram Murthy
EuMI Visiting Scholar CSE, IIT Madras, India

Mahesh Marina
School of Informatics University of Edinburgh, UK

***Abstract*—Efficient utilization of Multi Channel - Multi Radio (MC-MR) Wireless Mesh Networks (WMNs) can be achieved only by intelligent Channel Assignment (CA) and Link Scheduling (LS). Due to the dynamic nature of traffic demand in WMNs, the CA has to be reconfigured whenever traffic demand changes, in order to achieve maximum throughput in the network. The reconfiguration of CA requires channel switching which leads to disruption of ongoing traffic in the network. The existing CA algorithms for MC-MR WMNs in the literature do not consider the channel reconfiguration overhead that occurs due to this channel switching. In this paper, we propose a novel reconfiguration framework that considers both network throughput and reconfiguration overhead to quantitatively evaluate a reconfiguration algorithm. Based on the reconfiguration framework, we propose an online heuristic algorithm for CA called Demand based State Aware channel Reconfiguration Algorithm (DeSARA) that finds the CA for the current traffic demand by considering the existing CA of the network to minimize the reconfiguration overhead. We show through simulations that DeSARA outperforms both static CA and fully dynamic CA in terms of total achieved throughput.**

I. Introduction

Wireless Mesh Networks (WMNs) have become a cost-effective option for wide scale deployment in last mile wireless networks [1]. WMNs consist of two kinds of network elements namely, mesh routers and mesh clients. While the mesh routers form the backbone, the mesh clients are the users that generate traffic in the network. Apart from clients communicating with the Internet, there are a number of user applications such as video conferencing and VoIP (Voice over IP) that result in communication between clients. As more and more clients are added to WMNs, there is a requirement to improve the transport capacity of the backbone.

The transport capacity of the backbone network can be increased by using multiple orthogonal (non-overlapping) channels for simultaneous transmissions and thereby, improving *channel spatial reuse*. To tap the full potential of multiple channels, the mesh routers are equipped with multiple radios. Raniwala *et al.* [2] showed that there is a non-linear increase in capacity with the increase in number of radios in a Multi Channel - Multi Radio (MC-MR) wireless network. Though there is a potential increase in capacity due to the usage of MC-MR in WMNs, a poor Channel Assignment (CA) scheme can lead to under-utilization of the network. Intelligent CA schemes need to be adopted in order to spatially separate the nodes transmitting in the same channel as far as possible. Nevertheless, owing to the constraint on the number of channels and the number of radios available at each router, the co-channel interference cannot be completely avoided.

A network employing multiple access techniques such as CSMA/CA for transmission of data is not bandwidth efficient due to the contention resolution mechanism employed for the channel access. Using efficient Link Scheduling (LS), the network can be made contention free thus utilizing the capacity of the network completely. This requires a link level synchronization among the nodes. As the mesh routers are static, link level synchronization between them can be achieved easily.

The distributed algorithms for CA and routing [3], [4], [5] improve the network performance in terms of throughput but they are not optimal in terms of offered load. There are several works on interference aware CA aimed at minimizing the bandwidth lost in collisions [6], [7], [8]. All these works do not take into account the dynamic nature of the traffic demand in the network and hence these solutions cannot utilize the network resources efficiently. Considering the traffic demand in finding CA and LS in a MC-MR WMNs is beneficial in terms of achieved throughput as shown in [9], [10]. In a practical scenario, the traffic demand keeps changing with time and the WMN must be able to adapt to the changes by modifying the bandwidth allocation to the links. Under dynamically varying traffic conditions the CA designed to optimally suit one traffic demand may not optimally suit the other. If we use the optimal CA for every traffic demand in the network, without considering the current state of the network, it will lead to lot of reconfigurations and will heavily disrupt the ongoing traffic in the network. In [11], the authors have shown that though the channel switching time of the interface cards are in the order of micro seconds, the average traffic disruption time due to channel reconfiguration is in the order of seconds. If these reconfigurations happen frequently, the loss of packets and under utilization of the network will make WMNs very unreliable. Hence, there is a need for simultaneously improving the utilization of the network resources and reducing the disruption of the traffic due to reconfiguration of CA.

To the best of our knowledge, none of the works in the literature considers the reconfiguration overhead in CA. In this work, we propose a novel reconfiguration framework to quantitatively evaluate a reconfiguration algorithm and show the significance of reconfiguration overhead in CA. This reconfiguration framework considers both the achieved throughput and the reconfiguration overhead to quantify the performance of a reconfiguration algorithm. We also present an online heuristic algorithm called Demand based State Aware channel Reconfiguration Algorithm (DeSARA) for dynamically reconfiguring the CA under varying traffic conditions. We show the performance of DeSARA in different network and traffic scenarios using simulations.

II. System Model and Assumptions

In this paper, we limit our study to the mesh routers which form the backbone. Hereafter, the term node is used to refer to a mesh router. Similarly, we use the terms traffic matrix and traffic demand interchangeably throughout this paper.

A. Network Model

The network is represented in the form of an *undirected graph* $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{n_1, n_2, \ldots, n_V\}$ represents the set of nodes in the network and $\mathcal{E}$ represents the set of wireless links. G represents the topology of the network, having all *potential* transmission links represented by edges. Each node n_i has K_{n_i} radios installed in it. These radios can operate simultaneously independent of each other. Logically, to operate them simultaneously, they need to be tuned to *orthogonal* channels. We assume that $\mathcal{C} = \{c_1, c_2, \ldots, c_C\}$ be the set of orthogonal channels available in the network and their individual maximum capacity as C_{max}. We denote by V, E, and C the cardinality of the sets $\mathcal{V}$, $\mathcal{E}$, and $\mathcal{C}$, respectively. Note that if $C < \underset{n \in V}{max}(K_n)$, then some radios can not be used, hence we assume that $C > \underset{n \in V}{max}(K_n)$. The number of available orthogonal channels depends on the radio frequency spectrum used. IEEE 802.11a and IEEE 802.11g support 12 and 3 orthogonal channels, respectively. An edge $e \in \mathcal{E}$ exists between a node pair i, j iff the nodes i and j are in the transmission range of each other. The assumption of an undirected graph implies that the transmission range of the radios is equal. There are a total of T slots for transmission and the links must be scheduled to transmit in these slots. We denote the slot schedules by $X^t_{e,c} \in \{0, 1\}$. $X^t_{e,c} = 1$ if an edge $e \in \mathcal{E}$ is assigned a slot $t \in \{1, 2, \ldots, T\}$ in the channel $c \in \mathcal{C}$. Let F_G be the *Conflict Graph* of G. Each edge e in G is represented by a node m_e in F_G. An edge exists between nodes $m_e, m_{e'}$ in F_G if $e, e' \in \mathcal{E}$ in G interfere with each other (based on the interference model considered). The adjacency matrix representation of the graph F_G is called as Link Interference Matrix (LIM) [7]. $LIM_{e,e'}$ denotes if edges e and e' interfere or not. We define a binary vector $W_{e,c}$ to denote the channel assignment at edges. $W_{e,c}$ is 1 if channel c is assigned to the edge e. Similarly, we define a binary vector $Y_{n,c}$ to denote the channel assignment at nodes. $Y_{n,c}$ is 1 if channel c is assigned to one of the radios of node n.

B. Traffic Model

The traffic demand is modeled as a matrix $\mathcal{T}$ in which each entry t_{ij} represents the traffic (data rate) between the node pair i, j. We assume that a source-destination pair (node pair) can transmit data in more than one alternate path simultaneously and that the traffic is *infinitely divisible*. The alternate paths between every source-destination pair, discovered by the routing algorithm, are given by $f^e_{ij,k} \in \{0, 1\}$ where $i, j \in \mathcal{V}$, $e \in \mathcal{E}$, and k denotes the k^{th} path between node pair i, j. If the k^{th} path between node pair i, j passes through the edge e then the variable $f^e_{ij,k} = 1$ else $f^e_{ij,k} = 0$. The *throughput* between node pair i, j (denoted by x_{ij}) is given by $\sum_k x_{ij,k}$ where $x_{ij,k}$ is the achieved throughput on the k^{th} path between node pair i, j. The alternate paths are precomputed by k-shortest paths algorithm [12].

C. Assumptions

We assume the existence of a *traffic profiler* at every mesh node that observes the current traffic (data rate) for each destination in the network at a fixed interval and communicates the information to a centralized agent [6]. We also assume that link level synchronization exists among the nodes. Due to the static nature of the WMNs and the presence of a centralized agent, link level synchronization can be achieved easily. Many works in WMNs have assumed the existence of link level synchronization among the nodes [4], [9].

D. Reconfiguration Framework

The network utilization after a reconfiguration due to the arrival of a traffic matrix can be captured by the achieved throughput (Tp) and is represented by $Tp = \sum_{1 \le i < j \le V} \sum x_{ij}$.

Let $Pr_{ij,k} = 1$ if an edge on the k^{th} path between nodes i, j is reconfigured, otherwise $Pr_{ij,k} = 0$. Then the reconfiguration overhead ($\mathcal{R}$), which is the sum of flows that are disrupted due to a reconfiguration, is represented by $\mathcal{R} = \sum_{1 \le i, j \le V} \sum \sum_k Pr_{ij,k} \times x_{ij,k}$.

Let α denote the time between arrival of two successive traffic matrices and β denote the time required to successfully complete a reconfiguration. The values of α and β can be obtained from the underlying network. The value of α solely depends on the kind of network traffic generated by the users. Whereas the value of β depends on factors such as the kind of hardware used (the switching time of the radios) and the mechanism used to transmit the reconfiguration information to the nodes. Then the amount of effective data transfered (EDT) by a reconfiguration algorithm for a traffic demand can be measured by $EDT = (\alpha \times Tp - \beta \times \mathcal{R})$.

III. Online Reconfiguration Algorithm

In this section, we propose a reconfiguration algorithm called Demand based State Aware channel Reconfiguration Algorithm (DeSARA) for dynamic adaptation to changing traffic demands. For every α units of time, DeSARA finds a new CA and LS for the current traffic demand considering the existing state (CA) of the network. The main objective of DeSARA

is to increase the total achieved throughput of the network by minimizing the reconfiguration overhead. DeSARA consists of three major steps namely Channel Assignment (CA), Link Scheduling (LS), and Flow Allocation (FA).

A. Channel Assignment

As mentioned earlier, the routing algorithm gives a set of alternate paths that are represented by the binary variables $f^e_{ij,k}$. Now, we have to find the amount of traffic that can be sent in each of these paths between every source-destination pair. The optimal flow allocation on these paths can be done only after CA and LS. But the CA is highly dependent on the FA. This leads to a cyclic inter-dependency between CA and FA. We break this inter-dependency by assigning probabilities to the routes (lower hop count must be given higher probability) which denote the likelihood of a particular route to be included in the final route selection. We represent the hop count of k^{th} path between i, j pair as $HopCount_{ij,k}$.

The probabilities $\mathcal{P}_{ij,k}$ assigned to the paths are as follows:

$$\mathcal{P}_{ij,k} = \frac{\rho_{ij,k}}{\sum_{k'} \rho_{ij,k'}}, \text{ where } \rho_{ij,k} = \frac{1}{HopCount_{ij,k}} \quad (1)$$

Based on these probabilities the expected load on each path and the expected load on each edge can be calculated as follows:

$$Exp_{Path}(ij,k) = \mathcal{P}_{ij,k} \times t_{ij} \quad (2)$$

$$Exp(e) = \sum_{1 \le i < j \le V} \sum_{k} (f^e_{ij,k} \times Exp_{Path}(ij,k)) \quad (3)$$

We define another useful quantity $Exp_{Reg}(e)$ for each edge $e \in \mathcal{E}$ which captures the expected load in the interference region of edge e. This consists of the expected load on edge e and on all the edges $e' \in \mathcal{E}$ that could potentially interfere with edge e.

$$Exp_{Reg}(e) = Exp(e) + \sum_{e' \in \mathcal{E}} (LIM_{e,e'} \times Exp(e')) \quad (4)$$

The effective data transmitted on an edge e during α units of time if it is assigned channel c is as follows:

If there is no reconfiguration

$$EffecData(e,c) = \frac{\alpha \times C_{max} \times Exp(e)}{Exp_{Chan}(e,c)} \quad (5)$$

else if there is reconfiguration $EffecData(e,c)$

$$= \frac{\alpha \times C_{max} \times Exp(e)}{Exp_{Chan}(e,c) + Exp(e)} - \beta \times Reconf(e) \quad (6)$$

where $Exp_{Chan}(e,c)$ is the expected load on channel c at edge e (i.e., sum of the expected loads on edges in the interference region of edge e which are assigned channel c). $Reconf(e)$ is the total amount of flow that will be affected if this edge is assigned a new channel and is given by

$$Reconf(e) = \sum_{1 \le i,j \le V} \sum_{k} x^{old}_{ij,k} \times f^e_{ij,k},$$

where $x^{old}_{ij,k}$ is the existing traffic on the k^{th} path between node pair i, j.

Algorithm 1 Channel Assignment Algorithm

Input: A Graph $G = (\mathcal{V}, \mathcal{E})$, $Exp(e)$, $Exp_{Reg}(e)$, for each link $e \in \mathcal{E}$ and $W^{curr}_{e,c}$ - the existing channel assignment
Output: The New Channel Assignment for edges, $W^{new}_{e,c}$
Set $W^{new}_{e,c} \leftarrow 0 \quad \forall e \in \mathcal{E},\ c \in \mathcal{C}$ */*Initialize the $W^{new}_{e,c}$ variables */*
Let $\mathcal{E}' \leftarrow \mathcal{E}$, $SatNodes \leftarrow \Phi$
while $\mathcal{E}'$ is not empty **do**
  if $SatNodes$ is empty **then**
    Find the edge $e \in \mathcal{E}'$ in G with largest $Exp_{Reg}(e)$. In case of a tie choose the one with high $Exp(e)$.
    Find the $Exp_{Chan}(e,c)$ at edge e
    Find the $EffecData(e,c)$ for each channel at edge e
    Choose c with the maximum $EffecData(e,c)$. In case of tie choose c with minimum $Exp_{Chan}(e,c)$.
    $c_x \leftarrow c$, $W^{new}_{e,c_x} \leftarrow 1$, $\mathcal{E}' \leftarrow \mathcal{E}' - \{e\}$ */*Assign channel c_x to e*/*
    $Y_{i,c_x} \leftarrow 1$, $Y_{j,c_x} \leftarrow 1 \ \ where\ e = (i,j)$ */*Assigning the chosen channel to the incident nodes */*
    */*Check for saturation of radios on the incident nodes*/*
    if $\sum_{c'=1}^{C} Y_{i,c'} = K_i$ **then**
      $SatNodes \longleftarrow SatNodes \cup \{i\}$
    end if
    if $\sum_{c'=1}^{C} Y_{j,c'} = K_j$ **then**
      $SatNodes \longleftarrow SatNodes \cup \{j\}$
    end if
  else $\{SatNodes$ is not empty$\}$
    */*Assign the remaining edges of the currently saturated nodes with the last channel chosen */*
    while $SatNodes$ is not empty **do**
      Set $i \leftarrow GetOneElement(SatNodes)$
      Construct $UnassignedEdges(i) = \{e | e \in \mathcal{E}, i \in Inc(e), W^{new}_{e,c} = 0\ \forall c \in \mathcal{C}\}$
      for all $e \in UnassignedEdges(i)$ **do**
        */*Assign the last assigned channel c_x to the other edges*/*
        $W^{new}_{e,c_x} \leftarrow 1$, $SatNodes \leftarrow SatNodes - \{e\}$, $Y_{i,c_x} \leftarrow 1$, $Y_{j,c_x} \leftarrow 1$ where $e = (i,j)$
        */*Check if the other incident node is saturated*/*
        if $\sum_{c'=1}^{C} Y_{j,c'} = K_j$ **then**
          $SatNodes \longleftarrow SatNodes \cup \{j\}$
        end if
      end for
      $SatNodes \leftarrow SatNodes - \{i\}$
    end while
  end if
end while

The CA algorithm is given in Algorithm 1 whose computational complexity is $O(E^2C)$. The underlying logic of our CA algorithm is to try to minimize the co-channel interference among edges with high expected load.

We construct a set $\mathcal{E}'$ which is initialized to $\mathcal{E}$ and a set $SatNodes$ which is initialized to empty. As the edges are assigned channels, they are removed from the set $\mathcal{E}'$. This is repeated until $\mathcal{E}'$ is empty ($i.e.$, all the edges are assigned a new channel). For channel assignment, we choose the edge e with the highest $Exp_{Reg}(e)$. In case of a tie, we choose the edge with the highest expected load. The reason for using $Exp_{Reg}(e)$ as a criterion for choosing edge is that, it represents the edge with the highest potential to interfere with other edges. After choosing an edge for channel assignment, we assign a channel with the highest value of $EffecData(e,c)$ to that edge. In case of a tie, the channel with least value of $Exp_{Chan}(e,c)$ will be chosen. Once the channel is assigned to an edge, the last radio in either or both incident nodes may be utilized and the node(s) is(are) called *saturated*. We add a saturated node to the set $SatNodes$. In any iteration, if the $SatNodes$ is non empty, then there

are some nodes that became *saturated* in the previous step. The links incident on these saturated nodes that are yet to be assigned a channel, cannot be assigned a new channel due to the lack of a radio. So, the last assigned channel (in the previous iteration, which led to the saturation of the node) is then assigned to all those links.

B. Link Scheduling

Once the channels are assigned for current traffic demand, LS must be done for efficient resource sharing. We provide a greedy heuristic algorithm for the LS. The slots must be allotted in such a way that any two interfering links sharing a common channel must not be given the same slot. Hence, the total of T slots must be divided among the interfering links sharing the same channel. The division can be made on the basis of expected loads calculated for each edge. We calculate a tentative fraction of slots that needs to be provided to each edge in the following manner:

$$Frac(e) \quad = \quad \frac{Exp(e)}{\sum\limits_{e' \in \mathcal{E}} (LIM_{e,e'} \times W_{e',c} \times Exp(e'))}, \quad (7)$$

where $e \in \mathcal{E},\ c \in \mathcal{C},$ and $W_{e,c} = 1$

Algorithm 2 Link Scheduling

Input: LIM, $W^{new}_{e,c}$, $Frac(e)$, T and conflict graph F_G
Output: The slot schedules, $X^t_{e,c}$
Sort the edges in the decreasing order of the interference degree in their conflict graph F_G. Let $(e_1, e_2, \ldots, e_E)$ denote the sorted order
for $i = 1$ to E **do**
    $N(e_i) = ceil(T \cdot Frac(e_i))$, the maximum number of time slots e_i will be active
    Let $e_i = (u, v)$. $allocated \leftarrow 0,\ t \leftarrow 0$.
    Let $c \in \mathcal{C}$ be the channel allocated to edge e_i
    while $allocated \leq N(e_i)$ and $t \leq T$ **do**
        if $X^t_{e',c} = 0\ \ \forall e' \in \mathcal{E}$ and $LIM_{e,e'} = 1$ **then**
            $X^t_{e,c} \leftarrow 1,\ allocated++,\ t++$
        end if
    end while
end for

The number of slots can be calculated by $N(e) = \lceil T \times Frac(e) \rceil$. Since we use a ceiling function, this indicates the maximum number of slots that can be alloted to link e. Our LS algorithm is presented in Algorithm 2 and its computational complexity is $O(ET)$. The edges are sorted in the descending order of their interference degree in their conflict graph F_G. We start allotting slots to the edge with the highest degree. For each edge e, we check its interfering edges for free slots and assign them to the current edge e. This process continues until slots are alloted to all the edges.

C. Flow Allocation

Now, we need to do flow allocation in order to decide the amount of traffic to be sent on each of the alternate paths that are provided by the routing algorithm. The capacity $Cap(e)$ of an edge is the total available bandwidth alloted to the edge by LS and is given by

$$Cap(e) = \sum_{c \in \mathcal{C}} \sum_{t \in \{1,2,\ldots,T\}} \frac{X^t_{e,c}}{T} \times C_{max} \quad (8)$$

At each edge e, we find the bandwidth share for each flow passing through that edge using the expected load on each path given in Eq. 2.

$$Share(e, i, j, k) = \frac{Exp_{Path}(i, j, k)}{Exp(e)} \times Cap(e) \quad (9)$$

The bottleneck capacity ($Bottleneck(i, j, k)$) on a path k for a source-destination pair i, j is the minimum of residual capacities of all the edges in path $Path(i, j, k)$. Based on this $Bottleneck(i, j, k)$, the required bandwidth for each flow is alloted to all the edges on that path. Once the bandwidth is alloted to the flows in each edge, there may be unalloted bandwidth available between a source-destination pair i, j. This can be calculated by $t_{ij} - AllocatedFlow_{ij}$. The augmenting paths for each source-destination pair i, j can be used to find the unallocated bandwidth.

$$AugPaths_{ij} = \{< ij, k > | \\ f_{ij,k} = 1 \wedge Bottleneck(ij, k) > 0\} \quad (10)$$

The remaining bandwidth is alloted to source-destination pairs that can accommodate more traffic. The flow allocation algorithm is given in Algorithm 3 and its computational complexity is $O(E^2V^2)$.

Algorithm 3 Flow Allocation

Input: The slot allocation $X^t_{e,c}$ from the link scheduling algorithm
Output: The flow allocation between each node $AllotedFlow_{ij}$
Calculate the capacity $Cap(e)$ of each edge from $X^t_{e,c}$
Calculate the share of each flow on each of the edge $Share(e, ij, k)$
Initialize Residual Capacity $ResCap_e$ of each edge to $Cap(e)$
Let $\mathcal{E}' \leftarrow \mathcal{E}$
while $\mathcal{E}'$ is not empty **do**
    Find the edge $e \in \mathcal{E}'$ in G with highest $Exp(e)$.
    $FlowAllocable \leftarrow min(Share(e, ij, k), t_{ij}, Bottleneck(ij, k))$
    /*Updating the residual capacity of the edge and the flow allocated*/
    $ResCap_e \leftarrow ResCap_e - FlowAllocable$
    $AllocatedFlow_{ij} \leftarrow AllocatedFlow_{ij} + FlowAllocable$
end while
/*Find if there are any augmenting paths*/
$AugPathSet \leftarrow \{< i, j > | \exists AugPaths_{i,j} \wedge t_{ij} - AllocatedFlow_{ij} > 0\}$
while $AugPathSet$ is not empty **do**
    Find tuple $< i, j >$ in $AugPathSet$ with maximum $t_{ij} - AllocatedFlow_{ij}$
    while $\exists$ an element $< ij, k >$ in $AugPaths_{ij}$ **do**
        $AllotedFlow_{ij} \leftarrow AllotedFlow_{ij} + Bottleneck(ij, k)$
        $\forall e \in Path(ij, k) ResCap_e \leftarrow ResCape - Bottleneck(ij, k)$
    end while
    $AugPathSet \leftarrow AugPathSet - tuple < ij >$
end while

IV. Simulation Results

In this section, we study the performance of our proposed online reconfiguration of CA and LS algorithm (DeSARA) using simulations. We compare DeSARA with *Static Channel Assignment (SCA)* and *Dynamic Channel Assignment (DCA)* to show the importance of considering current CA in channel reconfiguration. In SCA, we use the distributed channel assignment algorithm SAFE (Skeleton Assisted partition FrEe) [3] to find the channel assignment for the network and use the link scheduling and flow allocation algorithms proposed for DeSARA. SAFE is a load unaware channel assignment algorithm which finds a feasible channel assignment ensuring the

connectivity of the network. In DCA, we use the same channel reconfiguration algorithm in DeSARA without considering the reconfiguration overhead while choosing the best channel. So, we calculate the $EffecData(e,c)$ based on Eqn. 5 for all the channels.

A. Generation of Traffic Sequences

As the correlation between successive traffic demands affects the performance of reconfiguration algorithms, we propose a traffic generation model to generate correlated traffic matrices. We generate a traffic sequence by initially generating a random traffic matrix and then generate the subsequent traffic matrices in the following manner:

- $\rho_1 \in [0,1]$ fraction of flows (randomly chosen among $\binom{V}{2}$ flows) are changed in subsequent traffic matrices.
- The selected flows are perturbed by a fraction $\rho_2 \in [0,1]$ from their previous value.

The set of traffic matrices generated in this manner exhibit some correlation between consecutive traffic matrices. Lower values of ρ_1 and ρ_2 imply higher correlation between the consecutive traffic matrices. All these traffic matrices are normalized to 1. The *load* of a given traffic demand is the sum of traffic between all the source-destination pairs ($\sum_{1 \le i < j \le V} \sum t_{ij}$). So, the traffic sequence for a specific load is obtained by generating a traffic sequence as mentioned above and scaling each of the traffic matrices to the required value of load.

B. Simulation Parameters

We study the performance of our proposed online heuristic algorithm in two different network topologies namely, a regular topology (7×7 grid with $150m$ between nodes in vertical and horizontal direction) and a random topology (50 nodes uniformly placed in $1000m \times 1000m$ terrain). The transmission range of each node is set to $200m$. The topology of the network is formed by adding an edge between two nodes iff they are in the transmission range of each other. The LIM is generated using the protocol interference model. We generate 5 paths of the least hop count between any two node pairs in the network using the multi-path routing algorithm given in [12]. The various parameters that are relevant to the simulation are given in Table I. The simulation results are obtained for 15 different runs and plotted with a confidence level of 95%.

TABLE I
PARAMETERS USED IN THE SIMULATION

Parameter	Value	Parameter	Value
Number of Radios (K)	2 (Grid), 5 (Random)	Number of Slots	30
Number of Channels (C)	3 (Grid), 6 (Random)	Traffic Matrices	300
Channel Capacity (C_{max})	11 Mbps	ρ_1,ρ_2	0.3,0.3
α	100secs	β	1sec

C. Throughput with Traffic Matrices

To show that the reconfiguration algorithm reduces the overhead, we measured both throughput and reconfiguration overhead for each traffic matrix in regular topology and is shown in Fig. 1 and Fig. 2, respectively. It is clear from Fig. 1 that the throughput achieved by DCA, and DeSARA is significantly higher compared to that of SCA. This is due to the fact that the SCA does not consider the current traffic demand. It shows the importance of considering the traffic demand in CA. Fig. 2 shows that the reconfiguration overhead is reduced significantly by DeSARA as it tries to minimize the number of reconfigurations in the network. As DCA does not consider the existing CA and tries to find the best CA for the network, it incurs more reconfiguration overhead. Fig. 3 shows the aggregate data transfered with the traffic matrices. DeSARA performs better than both DCA and SCA.

D. Total Achieved Throughput with Load

Fig. 4 and Fig. 5 show the total achieved throughput with load in regular and random topologies, respectively. As the offered load increases, the total achieved throughput linearly increases for low load and saturates at higher load. For a given network with a fixed number of radios and channels, DCA and DeSARA are able to achieve higher total throughput than SCA. The state aware reconfiguration algorithm DeSARA performs better than DCA due to the fact that it minimizes the reconfiguration overhead. Both regular and grid topologies show similar performance comparisons between these algorithms. Between the regular and random topology, the regular topology is able to achieve higher total throughput compared to random topology. In our simulation studies, we considered only 2 radios and 3 orthogonal channels for regular grid topology but 5 radios and 6 channels for random topology. The grid topology with less number of radios and channels is able to achieve the same performance as that of the random topology with more number of radios and channels. This shows the effect of topology on total achieved throughput.

E. Effect of Correlation between Traffic Matrices

To study the effect of correlation between traffic matrices, we measured the total achieved throughput for a sequence of 300 traffic matrices with different values of ρ_1 and ρ_2. With lower values of ρ_1 and ρ_2 there is a higher correlation between consecutive traffic matrices and vice versa. We varied both of them together to change the correlation between consecutive traffic matrices. For each value of ρ_1 and ρ_2, the total achieved throughput is measured for 15 different runs of all the algorithms in a regular grid topology. From Fig. 6, we can see a reduction in the total achieved throughput as the correlation between consecutive traffic matrices decreases (increase in ρ_1 and ρ_2). Here too, we can note that DeSARA performs better than SCA and DCA.

V. CONCLUSION

For a successful deployment of MC-MR WMNs, the mesh routers should be able to reconfigure according to the changing traffic demands. But, the adaptation to changes in traffic demand requires reconfiguration of CA and LS for the efficient utilization of network resources. The channel switching involved in the reconfiguration of CA, necessitate the need for an efficient reconfiguration algorithm that minimizes the reconfiguration overhead. In this paper, we proposed a framework

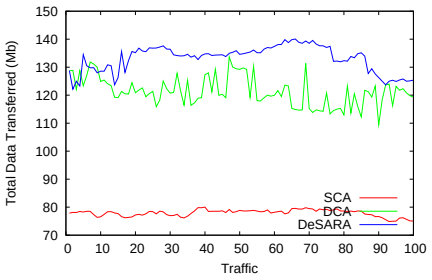

Fig. 1. Total Data Transfered with the Number of Traffic Matrices.

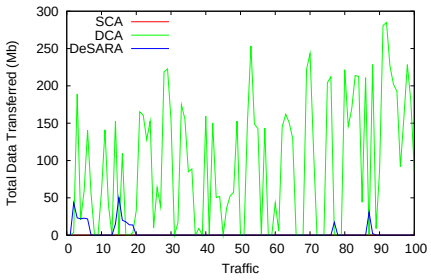

Fig. 2. Reconfiguration Overhead with the Number of Traffic Matrices.

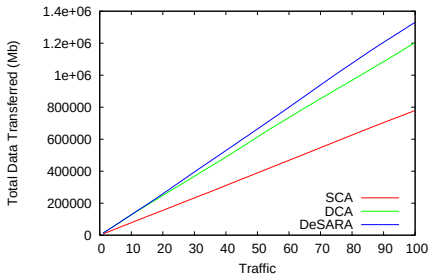

Fig. 3. Aggregate Date Transfered with the Number of Traffic Matrices.

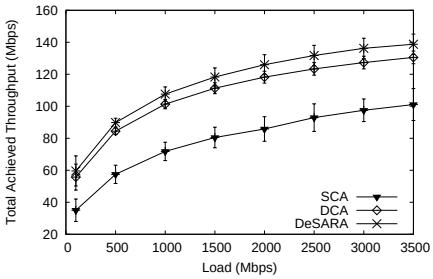

Fig. 4. Total Achieved Throughput with Load in a 7×7 Grid Topology.

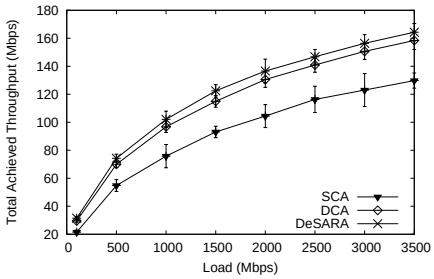

Fig. 5. Total Achieved Throughput with Load in a Random Topology of 50 Nodes.

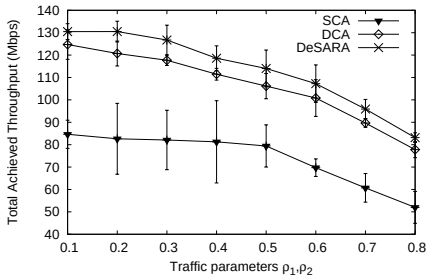

Fig. 6. Total Achieved Throughput with Different Values of ρ_1 and ρ_2 in a Grid Topology.

to evaluate a reconfiguration policy based on two conflicting objectives: maximizing network utilization and minimizing traffic disruption. We proposed a polynomially bound online heuristic algorithm called Demand based State Aware channel Reconfiguration Algorithm (DeSARA). DeSARA shows an improvement of about 15% in the total achieved throughput compared to DCA (DCA does not consider the reconfiguration overhead). This is mainly because DeSARA reduces the reconfiguration overhead by about 80%. We have shown through extensive simulations that DeSARA outperforms both SCA and DCA in terms of total achieved throughput. As part of future work, we plan to employ some traffic prediction techniques in order to reduce the frequency of channel reconfiguration. Also, we plan to implement our proposed reconfiguration algorithm in a real WMN testbed and evaluate its performance.

ACKNOWLEDGEMENT

This work was supported by the Department of Science and Technology, New Delhi, India.

REFERENCES

[1] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless Mesh Networks: A Survey," *Computer Networks*, vol. 47, no. 4, pp. 445–487, March 2005.

[2] A. Raniwala, K. Gopalan, and T. Chiueh, "Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 8, no. 2, pp. 50–65, 2004.

[3] M. Shin, S. Lee, and Y. Kim, "Distributed Channel Assignment for Multi-Radio Wireless Networks," in *Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS 2006)*, October 2006, pp. 417–426.

[4] X. Lin and S. Rasool, "A Distributed Joint Channel-Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad-hoc Wireless Networks," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM 2007)*, May 2007, pp. 1118–1126.

[5] H. Wu, F. Yang, K. Tan, J. Chen, Q. Zhang, and Z. Zhang, "Distributed Channel Assignment and Routing in Multiradio Multichannel Multihop Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 1972–1983, November 2006.

[6] K. N. Ramachandran, E. M. Belding, K. C. Almeroth, and M. M. Buddhikot, "Interference-Aware Channel Assignment in Multi-Radio Wireless Mesh Networks," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM 2006)*, April 2006, pp. 1–12.

[7] A. K. Das, H. M. K. Alazemi, R. Vijayakumar, and S. Roy, "Optimization Models for Fixed Channel Assignment in Wireless Mesh Networks with Multiple Radios," in *Proceedings of the IEEE Communication Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2005)*, September 2005, pp. 463–474.

[8] A. P. Subramanian, H. Gupta, and S. R. Das, "Minimum Interference Channel Assignment in Multi-Radio Wireless Mesh Networks," in *Proceedings of the IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2007)*, June 2007, pp. 481–490.

[9] H. Yu, P. Mohapatra, and X. Liu, "Channel Assignment and Link Scheduling in Multi-Radio Multi-Channel Wireless Mesh Networks," *Mobile Network and Applications*, vol. 13, no. 1-2, pp. 169–185, 2008.

[10] A. A. Kanagasabapathy, A. A. Franklin, and C. S. R. Murthy, "A Load Aware Channel Assignment and Link Scheduling Algorithm for Multi-channel Multi-radio Wireless Mesh Networks," in *Proceedings of the Internataional Conference on High Performance Computing (HiPC)*, 2008, pp. 183–195.

[11] P. Li, N. Scalabrino, Y. M. Fang, E. Gregori, and I. Chlamtac, "How to Effectively Use Multiple Channels in Wireless Mesh Networks?" *IEEE Transactions on Parallel and Distributed Systems*. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2008.256

[12] E. de Queirós Vieira Martins and M. M. B. Pascoal, "A New Implementation of Yen's Ranking Loopless Paths Algorithm," *4OR: A Quarterly Journal of Operations Research*, vol. 1, no. 2, pp. 121–133, 2003.