Adjacent Channel Interference Reduction in Multichannel Wireless Networks Using Intelligent Channel Allocation*

Technical Report (March 2009)

Vijay Raman
Dept. of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, IL, USA
vraman3@illinois.edu

Nitin H. Vaidya
Dept. of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, IL, USA
nhv@illinois.edu

ABSTRACT

The performance of a multichannel, multi-radio wireless network is often limited by interference due to concurrent transmissions on the same and adjacent channels. These interference effects may be either due to simultaneous traffic activity by the multiple radios within a node or due to transmission by neighboring nodes. In this paper, we discuss simple local-balancing and interference-aware channel allocation algorithms for reducing the overall interference in the network. We evaluate the relative performance of our algorithms using actual implementations on a multichannel, multi-radio testbed. Using overall network throughput as a metric, we show through experiments that the channel allocation that is aware of the node's transmission activity performs better than the simple local-balancing algorithm, irrespective of the number of channels used for allocation. Additionally, we show that the performance benefit of a interference-aware allocation over the local-balancing allocation improves as the number of flows in the network increases. However, our experiments also reveal that the benefit of using a interference-aware algorithm over the simple local-balancing algorithm reduces when we have more number of channels available for allocation when compared to the number of flows in the network.

1. INTRODUCTION

Due to the broadcast nature of the wireless medium, the capacity scaling of a wireless network is mainly limited by interference due to simultaneous transmissions on any given channel [1]. Approaches for improving system performance by reducing the interference effect in a multihop wireless network include orthogonalizing nearby transmissions across different frequencies, time, or code. Multichannel wireless networks, which has been gaining popularity more recently, provide a practical means for orthogonalizing transmissions on different frequencies. Several existing technologies support the notion of multiple channels. For instance, IEEE 802.11a specifies twelve non-overlapping channels in the 5 GHz band for communication [2]. Additionally, many practical protocols have been proposed recently in the literature for utilizing multiple channels simultaneously [3, 4, 5, 6, 7]. Because the wireless radios that are currently available can tune to only one channel at a time, most of the multichannel protocols propose to use multiple radios in a wireless host [8, 4, 9].

In [10], the authors have studied the capacity scaling of a multichannel, multi-radio wireless network. They show that the capacity of a multichannel wireless network with n randomly distributed nodes scales linearly with the number of channels when the ratio of the number of channels to the number of radios is of the order of $O(\log n)$. For practical networks, this would mean that we can achieve higher throughputs by utilizing many channels, but with only few interfaces[1] (when compared to the number of channels used) per node.

One of the important issues that need to be considered while designing a multichannel, multi-radio wireless network is the cochannel and adjacent channel interference effects due to the close proximity of the radios in a node. Additionally, there can be cochannel and adjacent channel interference due to transmissions from neighboring nodes. It has been shown in the literature that these interference effects can cause a significant throughput loss in the network [11, 12, 13, 14]. While the cochannel interference restricts the number of times a channel can be re-used within a neighborhood, the adjacent channel interference effects impose a restriction on the number of channels that can be used in the network [11]. Some of the past works, such as [15, 16] have shown that adjacent channel interference effects are common in technologies, such as IEEE 802.11b and 802.11a due to channel overlaps.

Several approaches have been proposed in the literature for minimizing the adjacent channel interference effects ranging from coordinating the multiple radios in the wireless node [17], and adjusting the antenna parameters and the filter characteristics [12] to using the channel overlaps for routing data across devices operating on non-overlapping channels [15, 18]. One of the popular approaches for mitigating the interference effects is to choose the transmission channels carefully by making sure that nearby links are on channels that do not interfere sufficiently. However, due to the dynamic nature of the links in a wireless network, the interference characteristics may vary [14], and therefore the channel allocation should be adaptable to these variations. A simple channel allocation that reduces the overall interference in the network will be to use only non-contiguous channels that do not overlap with each other (such as, channels 1, 6, and 11 in 802.11b or channels 36, 44, 52, etc. in 802.11a). However, as we show later, because we use fewer channels this technique may reduce the overall network throughput. It is, therefore, desirable to design a channel allocation algorithm that can use all of the available channels efficiently.

In this paper, we propose two channel allocation algorithms that can utilize all the available channels effectively, and show that we achieve a significant improvement in the network throughput by reducing both the cochannel and adjacent channel interference ef-

*This research was supported in part by NSF grant 06-27074

[1]The terms interfaces and radios are used interchangeably in this paper and mean the same.

fects. We evaluate our algorithms using experiments conducted using actual implementations on a multichannel, multi-radio testbed.

The remainder of the paper is organized as follows. In Section 2, we provide a background on the multichannel testbed that is used for our experiments. In Section 4, we present our local-balancing and interference-aware channel allocation algorithms, and compare their relative performance benefits in Section 5. In Section 6, we present a survey of existing literature on the various channel allocation algorithms for mitigating interference in multichannel, multi-radio wireless networks, and conclude our paper in Section 7.

2. TESTBED SETUP

There has been several mesh networks testbeds built as a part of academic projects, such as UCSB MeshNet [19], Net-X [4], and iMesh [20]. We found the architecture used in Net-X testbed [4, 5] appropriate for our experiments. In this section, we present a brief overview of the testbed. The multichannel, multi-interface, multi-hop wireless testbed consists of about 20 Soekris net4521 (www.soekris.com) boxes distributed across various offices in our lab. Each of the testbed node has a 133MHz microprocessor, a compact flash (CF) card slot, two PCMCIA slots, and one mini-PCI slot. We run Linux kernel 2.4.26-based operating system on each of these boards. For our experiments, we equip the test nodes with one mini-PCI and one PCMCIA wireless card. These wireless cards are based on Atheros chipsets and are driven by madwifi drivers. The cards are operated in the IEEE 802.11a mode and are capable of operating on all the twelve channels of 802.11a. The mini-PCI cards make use of a pair of external antennas, and the PCMCIA card has its own internal antenna for communication.

Among the two wireless cards, one is used for transmitting data and the other is used for receiving data. **The channel on which a node receives data is assigned based on a channel allocation algorithm (which is discussed in this paper) and is fixed for durations that are larger than a packet transmission time. The channel on which a node transmits depends on the receive channel of the intended neighboring node. The transmit interface is capable of switching across channels for this purpose.** When two neighboring nodes have the same receive channel, then their receive interface is also used for transmission between them (instead of the transmit interface). The channel on which a node receives data is communicated to the neighboring nodes using periodic broadcast messages. For the purposes of our experiments, we make sure that every node is aware of the receive channels of the nodes that are up to two hops away.

A modified AODV routing protocol is used for routing packets across the network. The modifications to the original AODV protocol involves finding a channel diverse route that avoids bottlenecks and reduces the expected transmission time. These modifications are incorporated in to the route metric, called MCETT (multichannel expected transmission time) [21].

2.1 System Architecture

In this section, we briefly describe the important aspects of the Net-X system architecture. Details are available in [4] and [5].

The system architecture has three major components:

1. *Channel abstraction layer:* This kernel component manages multiple channels and interfaces, and provides support for fast interface switching. This component is generic enough to support other multichannel protocols, and other interface capabilities, such as data rates and transmission powers. The channel abstraction layer abstracts the details of multiple channels and interfaces from the higher layers, and is controlled by "IOCTL" commands from the userspace daemon.
2. *Kernel multichannel routing support:* This component is used to provide kernel support for on-demand routing. The component informs the userspace daemon when a route discovery has to be initiated, and buffers data packets while the route discovery is pending.
3. *Userspace daemon:* The userspace daemon implements the less time-critical components of higher layer protocols (currently this is a multi-channel routing/channel assignment protocol). Most of the higher layer protocol functionality is implemented in this component.

The kernel components interact with the Linux TCP/IP implementation and the interface device drivers, while the userspace daemon is built using standard userspace networking libraries.

2.2 Channel abstraction layer

The channel abstraction layer (CAL) is implemented as a part of the bonding driver present in the Linux kernel. Its key components include:

1. *Unicast component:* Enables specifying the channel to use to reach a neighbor.
2. *Broadcast component:* Provides support for sending broadcast packets over multiple channels.
3. *Scheduling and queuing component:* Supports interface switching by buffering packets when necessary, and scheduling switching across channels. The CAL maintains a per-channel queue for this purpose. The packets are queued in the appropriate channel queue depending on the channel in which a packet has to be sent in the next hop. Additionally, the CAL also maintains some statistics such as, the channel usage, the current channel used by the transmit interface, number of packets and bytes transmitted, the number of switches performed by the transmit interface, and the few other metrics that are useful for routing.

In addition, the madwifi driver for Atheros-based NICs (which are used by the wireless nodes) has been modified to support faster channel switching.

3. PROBLEM MOTIVATION

To motivate our channel allocation problem, we present two experiments for characterizing the interference effects in a multichannel wireless network. Because the interference effects in a multichannel wireless network are well studied in the literature (See [11, 12, 13, 14, 15]), we discuss only a small set of results to identify the possible interference effects in our testbed.

We define the following notation for our experiments.

1. The term 'hop' is defined as follows: If two nodes can have a direct communication link between them, then they are said to be

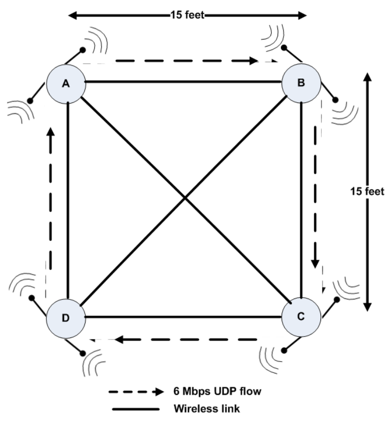

Figure 1. Four node network topology showing four adjacent flows.

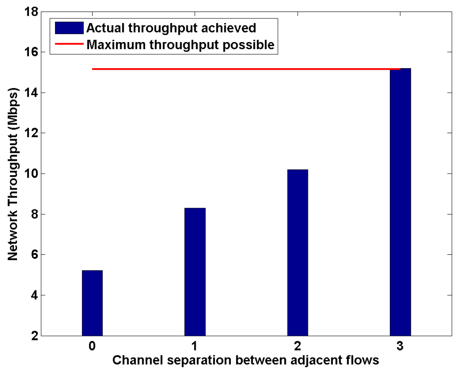

Figure 2. Throughput results for four flows in the form of ring.

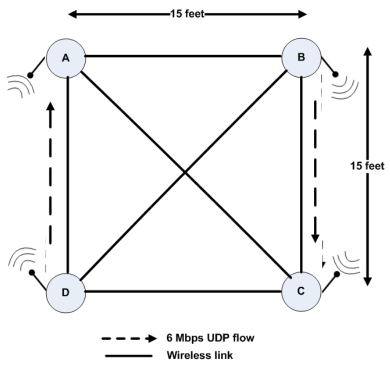

Figure 3. Four node network topology showing two non-adjacent flows.

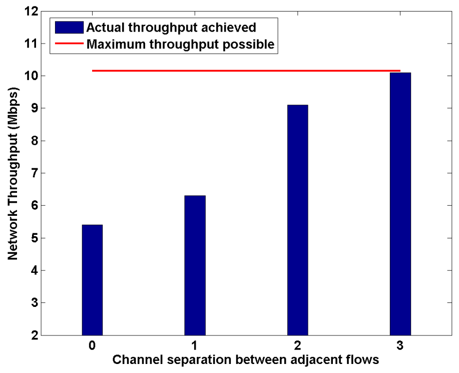

Figure 4. Throughput results for two non-adjacent flows.

within one hop from each other. If a transmission from one node to another requires k one hop transmissions, then the nodes are said to be k hops away from each other. **For the two experiments in this section, we use four nodes A, B, C, and D all of which are one hop from each other**.

2. Two single hop flows are said to be 'adjacent' if they share a node (either sending or receiving node).

3. IEEE 802.11a channels are numbered as 36, 40, 44, 48, 52, 56, 60, 64, 149, 153, 157, and 161. Let $\{c_0, c_1, \ldots, c_{11}\}$ denote the 12 channels listed above (so that, c_0 corresponds to channel 36, c_1 corresponds to channel 40, and so on) from which a receive channel is chosen for a node. **Suppose the nodes A, B, C, and D are all assigned the same receive channel (say c_1), then any data flow between the four nodes are considered to be separated by 0 channels. Suppose A, B, C, and D are assigned the channels $c_i, c_{i+1}, c_{i+2}, c_{i+3}$, respectively where $i \in \{0, 1, \ldots, 8\}$, then a flows AB and BC are separated by 1 channel. Similarly, flows from BC and CD are separated by 1 channel.** Note that flows CD and DA are separated by more than one channel. Because three out of four flows are separated by one channel, we refer to this channel allocation to be the case where adjacent flows are separated by one channel. In general, suppose the nodes A, B, C, and D are assigned the channels $\{c_i, c_{i+m}, c_{i+2m}, c_{i+3m}\}$, respectively (where $0 <= m <= 3$ in our experiments), then the flows from AB-BC and BC-CD are separated by m channels (ignoring the flows CD-DA).

In the first experiment, we generate four 6 Mbps unicast flows between the four nodes (A to B, B to C, C to D, and D to A) in the form of a ring, as shown by the dashed arrows (indicating the direction of flow) in Figure 1. This setup ensures that every node is both a source and a destination of a flow. We used the *iperf* utility, available in Linux, for generating the UDP traffic between the nodes. The physical transmission rate of the wireless cards are fixed at 6 Mbps and the UDP traffic is sent for a duration of 100 seconds. We choose the receive channels for the nodes so that any two adjacent flows are separated by 0 channels, and measure the throughputs achieved by the four nodes in each case. We then repeat this experiment by varying the receive channels of the nodes to separate the flows by 1, 2, or 3 channels (as measured previously), and measure the total throughput achieved by the four flows in each case. Figure 2 shows the total throughput values of the four flows (vertical bars) for the four different receive channel assignment, averaged over 30 runs. Additionally, we also measure the throughput of each of the flows separately (without transmitting the other flows) and plot their sum (shown as the horizontal line in Figure 2), again averaged over 30 runs (the receive channels allocated to the nodes do not matter in this case). We consider this as the maximum achievable throughput with this setup, and use this as a benchmark for comparing the throughputs when all the four flows are transmitted simultaneously. **We observe from the plot that the total throughput increases as the channel separation between the flows increases and there is no throughput loss compared to the maximum achievable throughput when the flows are separated by three channels.**

Next, we generate two non-adjacent 6 Mbps unicast UDP flows for 100 seconds as shown by the dotted arrows in Figure 3. One flow is directed from node D to node A, and the other from node B to node C. We then choose the receive channels for separating any two adjacent flows by 0, 1, 2 or 3 channels, as before and measure the total throughput achieved by the two transmissions. The throughput values averaged over 30 runs are shown in Figure 4. We also plot the sum of the throughputs achieved by the two flows when they are transmitted individially (horizontal line). **We observe from the plot that the trend is similar to the one observed in the previous case - the throughput improves as the channel separation between the flows increases. Furthermore, there is no significant throughput loss (when compared to the horizontal line) when the flows are separated by at least two channels.**

We have the following two observations from our experiments,

Observation 1: A simultaneous transmission and reception in a wireless node can interfere with each other if they are on channels separated by fewer than three.

Observation 2: Interference can also be due to transmission by a neighboring node, which is on a channel that is fewer than two channels away.

Based on these observations we impose the following constraints for a channel allocation to the nodes:

Constraint 1: No two interfaces (transmit or receive) on the same wireless node should operate simultaneously on channels that are separated by two or less. In our system, the channel on which a wireless node transmits depends on the receive channel of a neighboring node. Therefore, if a node is assigned the receive channel c_i, then the channels c_{i+1}, c_{i-1}, c_{i+2} or c_{i-2} should not be allocated to any of the one hop neighbors.

Constraint 2: None of the one hop neighbors of a node should transmit on the same or an immediately adjacent channel. Again, based on the fact that the channel on which a one hop node transmits depends on the receive channel of its immediate neighbors, we try to reduce the possibility of two hop neighbors of a node being allocated the same or an adjacent channel as the one allocated to the node under consideration. In other words, if a node is assigned the receive channel c_i, then the channels c_{i+1} and c_{i-1} should not be allocated to any of the two hop neighbors.

In the next section, we propose two channel allocation strategies for reducing the interference in the network with the objective of improving the overall network throughput. One of them does not use the constraints listed above and the other tries to implement the above constraints. We will later compare the relative performance of the two algorithms.

4. CHANNEL ALLOCATION ALGORITHM

Among the two algorithms proposed in this section, the first one, called LocBal is a simple local-balancing algorithm that balances the number of times a particular channel is used within a two hop neighborhood. A similar algorithm has been proposed by Kyasanur and Vaidya in [21]. The second algorithm proposes an interference-aware channel allocation mechanism, called IntAware in which the receive channel of a node is changed dynamically based on the traffic activity of the nodes and the constraints listed in Section 3.

4.1 The Local-Balancing Algorithm

According to this algorithm, every node first obtains the information on the receive channel assigned to all its one and two hop neighbors, `neighList`. The nodes can obtain this information by exchanging periodic broadcast messages as described in Section 2. Each node then counts the number of one and two hop neighbors, *chanCount*[*i*] that are assigned a particular channel, *i* and calculates the average, `mean`(*chanCount*) and the minimum, `min`(*chanCount*) utilization (number of nodes) of each of the channels. If `currChan` denotes the current receive channel of a node and `chanList` is the list of channels available for allocation, the local-balancing algorithm as executed periodically (every 5 seconds in our implementation) by every node, is as follows:

```
LocBal(currChan, chanList, neighList):
1. for $i$ in chanList
2.     $chanCount[i] \leftarrow 0$
3. for $neigh$ in neighList
4.     $chanCount[neigh \rightarrow channel]++$
5. if($chanCount[\mathtt{currChan}] > \mathtt{mean}(chanCount)$) AND
   ($chanCount[\mathtt{currChan}] > \mathtt{min}(chanCount) + 1$)
6.     With probability $p = \frac{1}{chanCount[\mathtt{currChan}]}$
7.         currChan ← minChan
8. return currChan
```

According to line 5 of the above algorithm, a node checks if the number of neighboring nodes that are on its current receive channel, *chanCount*[`currChan`] is more than the average utilization and at least one more than the minimum utilization over all channels in the network. If it is, then we probabilistically (with some probability $p = \frac{1}{chanCount[\mathtt{currChan}]}$) change the current receive channel of the node to the `minChan`, which is the channel with the minimum utilization. The reasoning for the condition in line 5 of the above algorithm is as follows: if *chanCount*[`currChan`] is less than or equal to the average utilization in the network, then there is no point in switching the receive channel to another channel as that will not balance utilization of all the channels in the network. On the other hand, if *chanCount*[`currChan`] is exactly equal to one more than the utilization of `minChan`, then the node will end-up switching its channel between `currChan` and `minChan`, which is not desirable. Finally, the probability $p = \frac{1}{chanCount[\mathtt{currChan}]}$, provides sufficient damping for the algorithm to converge.

Because LocBal balances the number of times any channel is used in the neighborhood, it reduces the cochannel interference in the network. However, this algorithm is agnostic to the adjacent channel interference effects. A naive approach for reducing the adjacent channel interference effects, as proposed by [21] is to use only five (non-contiguous) channels out of the 12 channels in IEEE 802.11a. However, we show through experiments that higher throughputs can be achieved if we use more contiguous channels than restricting the allocation to just the five non-contiguous channels.

For this experiment, we use a ten node network topology shown in Figure 5. We generate ten 6 Mbps one hop, unicast UDP flows in a ring fashion similar to the first experiment discussed in Section 3, so that every node is both a source and destination of one flow. This setup will ensure that every node undergoes the worst possible interference as both their radios are involved in some traffic activity (either reception or transmission). We compare the overall

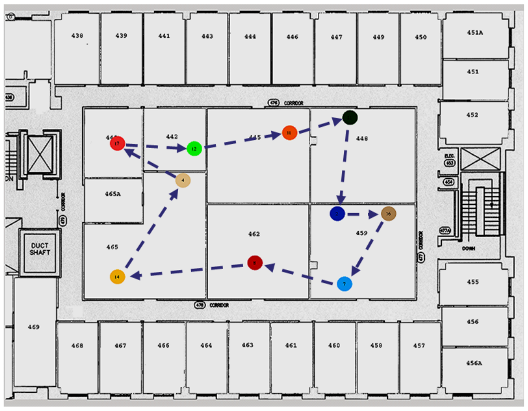
Figure 5. A ten node network topology- circles indicate nodes and their colors indicate their receive channels (10 different channels are shown); dotted arrows indicate a data flow directed from sender to receiver; links between nodes not shown for clarity.

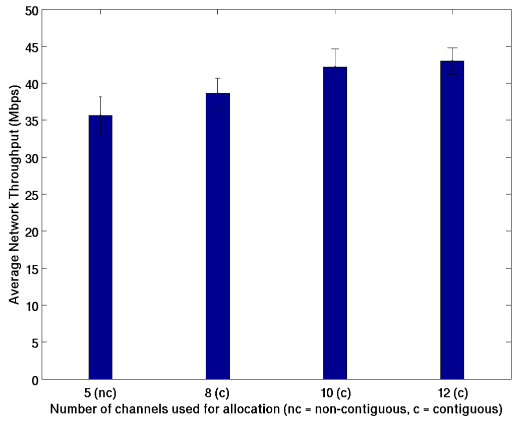

Figure 6. Throughput comparison for LocBal executed with 5 non-contiguous (nc) channels and 8, 10, 12 contiguous (c) channels.

network throughput (sum of individual throughputs of all the ten flows) achieved using the simple local-balancing allocation algorithm naively with just five non-contiguous channels (namely, 36, 48, 60, 149, and 161) with that of the same algorithm using 8, 10, and 12 contiguous channels. Thus, for the 8 channel case we use all the eight channels in the lower and middle UNII bands (namely channels 36, 40, 44, 48 ,52, 56, 60, and 64), for the 10 channel case we use two of the four channels (namely 153 and 157) in the upper UNII band in addition to the eight channels in the lower and middle UNII bands, and 12 channel case uses all the channels allowed in 802.11a.

Figure 6 shows the comparison plots of the network throughputs averaged over 30 runs. We let the auto rate functionality of 802.11 select the best possible physical transmission rates for the wireless cards. From the plots, we observe that the performance for the case where we use just 5 non-contiguous channels is worse than the case where we use 8, 10 or 12 contiguous channels. Furthermore, the throughput achieved is highest when we use all the twelve channels, as more channels ensure more concurrent, reliable transmissions in the network. From now on whenever we refer to LocBal, we mean to use contiguous channels, unless mentioned otherwise.

We wish to explore if a further improvement in throughput can be achieved over LocBal if adjacent channel interference effects are also taken care. Our interference-aware algorithm discussed in the next section adapts LocBal to address this point.

4.2 The Interference-Aware Algorithm

The interference-aware algorithm, in addition to reducing the cochannel interference effects by locally balancing the channels allocated, also reduces the adjacent channel interference effects both within a node and across the node by following the constraints in Section 3 as follows:

1. The adjacent channel interference within a node is reduced by ensuring that the transmit channel and the receive channel are separated by more than two. For this, we change the receive channel of a node when its transmit interface sends data on any of the channels that are within two channels away from the receive channel.

2. The adjacent channel interference across nodes is reduced by using a channel selection algorithm that allocates channels that are as spectrally farthest as possible. This reduces the possibility of two neighboring nodes being on channels that are separated by fewer than two channels. To determine the spectrally farthest channel from a channel i, we use the following distance factor,

$$distance_i = \sum_{\substack{j \in neighChannels, \\ j \neq i}} |i - j| \;\; \forall i \in K \tag{1}$$

where *neighChannels* is the set of channels assigned to the one and two hop neighbors of a node and K is the set of all channels that are considered for allocation. The spectrally farthest channel i is then given by $\arg\max_i \{distance_i\}$. In other words, for a given node we select a receive channel that is the farthest from all the channels allocated to its neighbors.

If `currChan` is the current receive channel of a node, and `chanList` and `neighList` contains the list of channels available for allocation and the list of one and two-hop neighbors of a node (along with their receive channels), then the interference-aware channel allocation and the channel selection algorithms, as executed periodically (every 5 seconds) by every node, is as follows:

```
INTAWARE(currChan, chanList, neighList):
1.   $M \leftarrow 100000$ //some large number
2.   for $i$ in chanList
3.       $chanCount[i] \leftarrow 0$
4.   for $neigh$ in neighList
5.       $chanCount[neigh \rightarrow channel]++$
6.   $currTxChanList \leftarrow$ GetInterfaceStats
7.   for $chan$ in $currTxChanList$
8.       $chanCount[chan \pm 1] \leftarrow M$
9.       $chanCount[chan \pm 2] \leftarrow M$
10.  $minChanList \leftarrow$ minChannels($chanList, chanCount$)
11.  if($chanCount$[currChan] $= M$)
12.      currChan $\leftarrow$ CHANSELECT(minChanList, neighList)
13.  else if($chanCount$[currChan] $\geq$ mean($chanCount$) $+ 1$) AND
     ($chanCount$[currChan] $>$ min($chanCount$) $+ 1$)
14.      With probability $p = \frac{1}{chanCount[\mathtt{currChan}]}$
15.      // create list of channels that have
16.      // the least channel count
17.      currChan $\leftarrow$ CHANSELECT(minChanList, neighList)
18.  return currChan
```

```
CHANSELECT(minchanList, neighList):
1.   for $i$ in minchanList
2.       for $neigh$ in neighList
3.           $distance[i] \leftarrow distance[i] + |i - (neigh \rightarrow channel)|$
4.   return maxDistChan
```

In interference-aware algorithm, in addition to counting the utilization of each of the channels based on the broadcast messages from the neighbors, every node also maintains a list of channels, *currTxChanList* on which it is currently transmitting or has a packet queued up for a future transmission. For this, we use a function, namely GetInterfaceStats, which obtains the information about the channels on which packets are currently transmitted or waiting to be transmitted from the wireless drivers. More information on the technique used for acquiring this information from the drivers is presented in Section 4.3. If the list is non-empty then the utilization of all the channels that are separated by fewer than three channels from any channel in *currTxChanList* is inflated artificially by a large number M. If *currTxChanlList* contains a channel separated by less than three from *currChan*, then *currChan* is immediately changed to a different channel as chosen by *ChanSelect* algorithm. Otherwise, the algorithm will behave similar to LOCBAL except for choosing the receive channel using the CHANSELECT algorithm.

To make sure that we do not select a channel whose *chanCount* is set as M, we form a potential list of channels that has the least utilization, namely *minChanList* using the function minChannels (see lines 10 and 11 in the INTAWARE algorithm). CHANSELECT then finds the channel that is spectrally farthest from all the channels in the neighborhood by evaluating the metric in Equation (1).

The INTAWARE algorithm, along with CHANSELECT algorithm, helps in reducing the adjacent channel interference effects form both within a node and from a neighboring node. Furthermore, because we also keep track of channels for which packets are awaiting to be sent in the near future, we reduce the possibility of switching to a channel that may be adjacent to the channel on which a future transmission may happen. This, in turn, reduces the number of times the receive channel is changed for a node thereby reducing the overhead involved in switching the receive channels. Though the interference awareness of this algorithm is restricted to traffic activity within a node, we show in the next section that the overall throughput achieved by the interference-aware algorithm is higher than the local-balancing algorithm.

4.3 Getting the Interface Statistics

The information on the channels on which the packets are currently transmitted or waiting to be transmitted is obtained from the channel abstraction layer (see Section 2.1). Because we need to communicate with a kernel module from a user-space program, we made use of a private IOCTL (input-output control) call to the kernel module from the user-space program. The relevant interface statistics information is obtained in the form of the estimated transmission time (ETT), which gives an estimate of the time taken by the packets to get transmitted on a link. In our testbed, we compute the ETT values for every channel queue maintained by the channel abstraction layer. The ETT for a channel queue is given by the following expression:

$$ETT = ETX * \frac{S}{B}$$

where, ETX is the expected number of transmission attempts (including re-transmissions) required to transmit a packet, S is the average packet size and B is the data rate of the link. The expected number of transmissions is estimated based on the loss in the link. A higher ETT value readily indicates that there is a packet waiting to be transmitted in the corresponding channel queue.

Whenever the INTAWARE channel allocation algorithm is executed, the algorithm sends an IOCTL query to the channel abstraction module requesting the ETT information of all the channel queues. The algorithm then checks if the ETT value of the channels adjacent to the current receive channel is above a certain threshold. If it is, then the algorithm searches for a lightly loaded channel that is spectrally farther from the other neighboring channels and switches to that channel. The channel is not switched if the ETT value is above the threshold in the current channel or in a channel other than the neighboring channels.

5. PERFORMANCE RESULTS

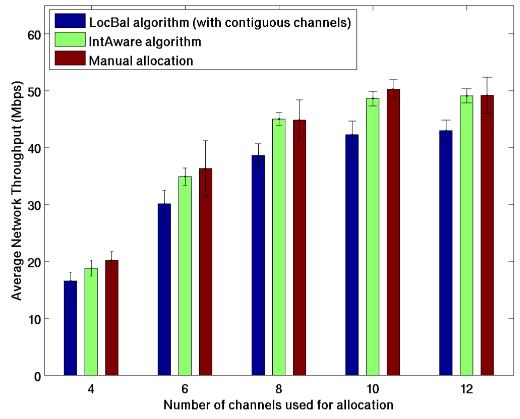

Figure 7. Comparison between the LOCBAL and INTAWARE algorithms - one transmission and one reception at each node.

We conduct two sets of experiments for evaluating the performance of our algorithms. The first set of experiments is for a deterministic

Table 1. Throughput (in Mbps) comparison between the LocBal and IntAware algorithms - one reception and one transmission at each node.

# Contiguous Channels	Manual Allocation (Mbps)	LocBal Algorithm (Mbps)	IntAware Algorithm (Mbps)	% Improvement IntAware over LocBal
4	20.15	16.53	18.77	13.56%
6	35.32	30.11	34.85	15.76%
8	44.86	38.62	44.99	16.52%
10	49.23	42.21	48.60	15.15%
12	48.17	42.98	49.06	14.14%

traffic pattern where the source and the destination nodes are determined a priori. Furthermore, we just generate one hop flows for the first set of experiments. Additionally, because of the deterministic nature of the traffic patterns generated, we also determine a manual channel allocation that is perceived to be good and use this as a benchmark for comparing our algorithms. We discuss more on the manual channel allocation in a follow-up section. For the second set of experiments, on the other hand, we choose the source and the destination nodes uniformly at random from the set of nodes and initiate multihop traffic between them. While the first set of experiments is useful in obtaining some interesting tradeoffs between our algorithms, the second set of experiments are intended to demonstrate the performance of our algorithms in a more realistic multi-hop scenario. Before we proceed to a discussion on our experimental results, we wish to explain more the manual channel allocation that were used as a benchmark for comparing our algorithms in the deterministic traffic pattern case..

Manual channel allocation used in our experiments

We obtain a channel allocation for the nodes manually by trying out many possible combinations of channel allocations to the nodes. Because the number of possible combinations of channel allocations is large, we tried out only those allocations that allocates non-adjacent channels to a one hop neighbor. In other words, having allocated a channel say, c, to a node, we ignore allocations that allocates channel $(c+1)$ or $(c-1)$ to any of the one hop neighbors of this node. For every allocation we averaged the overall throughput over 30 runs. **We then choose the allocation that resulted in the maximum overall throughput as a benchmark.**

Because of the processing constraint of the Soekris boxes, we only use UDP traffic at a fixed rate of 6 Mbps, for all our experiments discussed in this section. However, we let the auto rate functionality of the madwifi drivers to determine the best possible physical transmission rates.

5.1 Experiment Set 1 - Single-hop Deterministic Traffic Flows

5.1.1 One transmission and one reception per node:

We first discuss the experiments for a simple case where we have one transmission and one reception at each node. For this experiment, we use the same ten node network discussed in Section 4.1, shown in Figure 5. As shown in the figure, we generate ten 6 Mbps UDP flows such that every node is both a source and a destination of exactly one UDP flow. This ensures that both the radios are active in a node and therefore, creates the worst possible interference within a node. The transmission rate of the wireless cards is determined based on the auto rate functionality built in the drivers. We vary the number of contiguous channels available for allocation from 4 to 12, in steps of 2. **When the number of channels is 4, we use the channels 40, 44, 48, and 52, for 6 channels we use the channels 40 through 60. Similarly, for the case of 8 channels, we use channels 36 through 64. and for the 12 channel case we use all the 12 IEEE 802.11a channels allowed in the US.**

We repeated our experiment for five different traffic patterns by varying the source-destination pairs, and for each topology we measured the overall network throughput obtained using both the simple local-balancing and the interference-aware channel allocations. For each traffic pattern, we repeated our experiments for 30 runs. Note that Figure 5 shows only one of the five traffic patterns used in our experiments. The measured network throughput, averaged over all the traffic patterns and all the runs are shown in Figure 7. Figure 7 also shows the throughput obtained using the manual channel allocation. We have also plotted the 95% confidence interval bars for our throughput values. Table 1 tabulates the throughput values and the percentage improvement of interference-aware algorithm over the simple local-balancing algorithm.

From our figure, we first observe that the throughput achieved by our interference-aware algorithm is close to that achieved by a good channel allocation obtained manually. Additionally, we observe from the plots that the throughput obtained using IntAware is always better than that obtained using the LocBal algorithm. For instance, we can observe from Table 1 that using the interference-aware algorithm, a throughput improvement of up to 16.52% (corresponding to 8 channels) can be obtained over the local-balancing algorithm. However, we also observe that the throughput improvement obtained using the IntAware algorithm diminishes when we have fewer channels to allocate. In particular, when we use 4 channels, the throughput improvement reduces to 13.56%. This is due to the increased adjacent channel interference from the neighboring nodes when fewer channels are used for allocation. Additionaly, we can also observe that the throughput improvement achieved by IntAware is reduced when we have more number of channels when compared to the number of flows in the network. For instance, from Table 1, we observe that the throughput corresponding to the case of 12 channels is 14.14%. The reason for this is because, when we have more channels to allocate, even a local-balancing algorithm can achieve a channel allocation, such that the neighboring transmissions are on farther channels, which is good enough to reduce the interference in the network. This suggests that LocBal can be a simple alternative to the IntAware, when the number of channels available for allocation is larger than the expected number of flows in the network. However, as we can observe from the confidence interval bars, the variation in the throughput val-

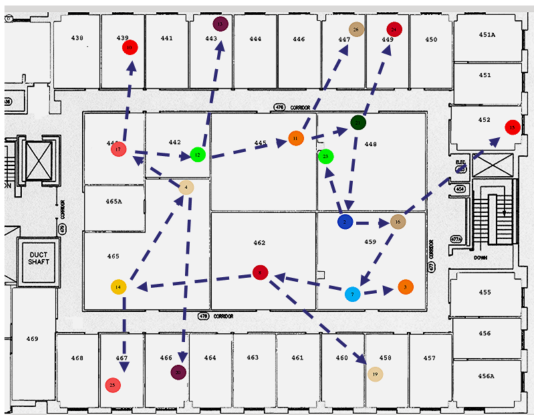

Figure 8. A twenty node network topology with 10 of the 20 nodes nodes having two outgoing flows (Please see Figure 5 for an explanation on the figure).

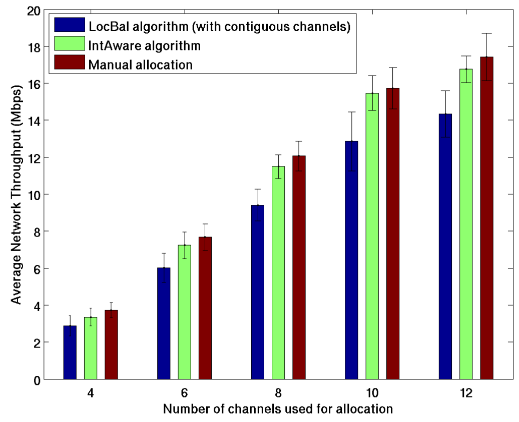

Figure 9. Comparison between the LocBal and IntAware algorithms - two transmissions and one reception at each node.

Table 2. Throughput (in Mbps) comparison between the LocBal and IntAware algorithms - two transmissions and one reception at each node.

# Contiguous Channels	Manual Allocation (Mbps)	LocBal Algorithm (Mbps)	IntAware Algorithm (Mbps)	% Improvement IntAware over LocBal
4	3.72	2.89	3.35	16.05%
6	7.67	6.03	7.24	20.09%
8	11.56	9.41	11.49	22.12%
10	15.73	12.86	15.47	20.34%
12	16.92	14.33	16.76	16.97%

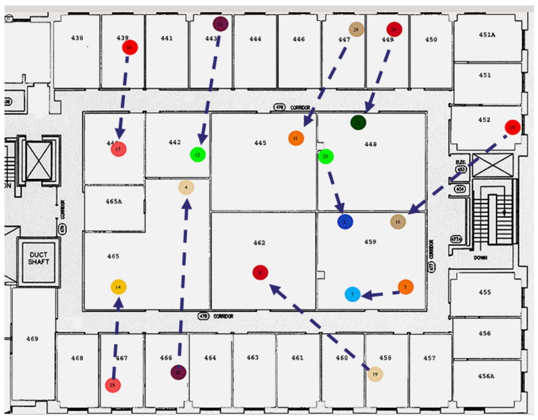

Figure 10. A twenty node network topology with each node involved in only one flow (Please see Figure 5 for an explanation on the figure).

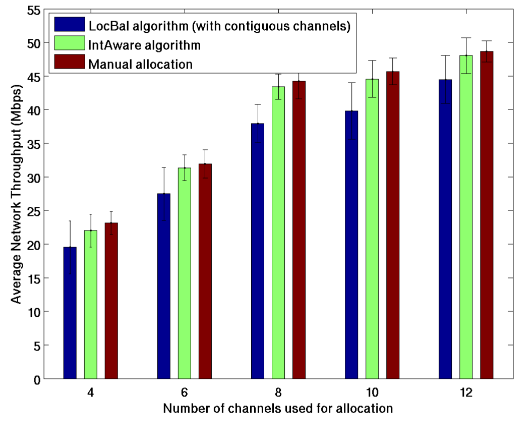

Figure 11. Comparison between the LocBal and IntAware algorithms - either one transmission or one reception at each node.

ues is higher for the local-balancing algorithm when compared to our trafic-aware algorithm. Moreover, the lower end of the confidence interval bar of the IntAware throughputs is above the upper end of the confidence interval bars of the LocBal throughputs in most cases. This shows that IntAware algorithm can consistently achieve higher throughputs when compared to IntAware.

5.1.2 More than one transmission and one reception at each node:

Next, we present the results for a scenario where a node is a source for more than one flow. In particular, in this experiment every node is a source for two flows, but is destination for exactly one flow. In this case, choosing a receive channel that does not interfere with both the transmissions from a node can be challenging. However, we show that the IntAware algorithm offers better throughputs

Table 3. Throughput (in Mbps) comparison between the LocBal and IntAware algorithms - either one transmission or one reception at each node.

# Contiguous Channels	Manual Allocation (Mbps)	LocBal Algorithm (Mbps)	IntAware Algorithm (Mbps)	% Improvement IntAware over LocBal
4	23.16	19.51	22.00	12.77%
6	31.92	27.47	31.33	14.07%
8	42.22	37.92	43.42	14.5%
10	41.67	38.76	42.34	9.24%
12	47.66	44.47	48.03	8.02%

even in this scenario. For this, we use the twenty node network shown in Figure 8 and generate five different traffic scenarios by choosing different source and destinations each time. One of the five different traffic patterns used for this experiment is also shown in Figure 8. According to this topology, ten out of twenty nodes have two outgoing flows and one incoming flow, each of which is a 6 Mbps UDP traffic, while the remaining ten nodes only have incoming flows. **Because we are only interested in the performance of the nodes that has both outgoing flows and incoming flows, we only measure the sum of the throughputs achieved by the nodes that send traffic**, averaged over the five traffic patterns, each repeated for 30 runs. The throughput values are shown in Figure 9, and are tabulated in Table 2. Note that we are not accounting for the throughputs at the nodes that do not have outgoing traffic from them as that may skew the observations. We once again observe from the figure that the IntAware algorithm always has a higher throughput than the LocBal algorithm. Additionally, the throughput values obtained using IntAware is closer to those obtained by the manual allocation. Furthermore, as we see from the tabulated throughput values that, though the actual throughput values are small owing to the increased interference in the network, the improvement in throughput obtained by IntAware over the LocBal algorithm is substantial than that in the previous experiment (where every node just had one outgoing flow). For instance, the improvement obtained using IntAware algorithm with 8 channels in this experiment is 22.12%, whereas that for the previous experiment with the same 8 channels is only 16.52%. This is because, when there are more flows in the network there is an higher chance for the flows to interfere with each other. The IntAware algorithm can successfully reduce the interference effects in the network in this case by choosing the best possible channel allocation. As in the previous experiment, we observe that the throughput improvement achieved using IntAware reduces when we have more number of channels available for allocation (compared to the ten flows whose throughput values are measured), suggesting that the IntAware algorithm can be used when we have more channels. Additionally, as in the previous experiment, we observe that the 95% confidence interval bars for the IntAware throughputs lie above the confidence intervals of LocBal throughputs suggesting consistent behavior of the IntAware algorithm.

5.1.3 Either one transmission or one reception at each node:

The previous two experiments showed that the interference-aware channel allocation is effective in reducing the interference due to simultaneous reception and transmission within a node that are on adjacent channels. When a node is not involved in a simultaneous reception and transmission, the overall interference in the network may be low. We wish to study the performance of the LocBal and IntAware algorithms for this scenario. For this case, we generate one hop UDP flows at a rate of 6 Mbps from each node to one of their immediate neighbor in a twenty node network as shown in Figure 10. Note that every node is involved either in a transmission or a reception, but not both. We then measure the total network throughput for five different combinations of source-destination pairs and plot the values after averaging them over 30 runs for each of the five combinations. Figure 11 shows the corresponding throughput values along with the 95% confidence intervals, and Table 3 tabulates the throughput values.

From the figure and the table, we observe that the throughputs achieved using IntAware algorithm is only slightly above those values achieved using the LocBal algorithm. This is because, the interference-awareness in IntAware algorithm is effective in reducing only the interference effects within a node, which is absent in this experiment. However, the improvement in throughput values of IntAware over LocBal in this case is due to the channel selection algorithm. The channel selection algorithm always chooses channels, within a neighborhood of nodes, that are spectrally farther apart thereby reducing the overall interference in the network. This shows that interference from neighboring nodes can be reduced by selecting as spectrally farther channel as possible. However, we observe from Figure 10 that the confidence intervals of the IntAware throughputs overlap with those of the LocBal throughputs in most of the cases. This suggests that further improvements can be made to the IntAware algorithm to provide better performance in this scenario. We wish to explore more on these aspects for our future work. One possibility for improving the throughput in this case will be to exchange the traffic information across nodes, as suggested in [22]. However, this might incur additional overhead in the network.

5.2 Experiment Set 2 - Multihop Random Traffic Flows

We now demonstrate the performance of our algorithms for a more realistic multihop network scenario. For this experiment, we use the same set of 20 nodes as before and choose 10 different source-destination pairs uniformly at random. The pairs of nodes can be situated anywhere in our network and therefore, are not necessarily one hop neighbors. We then generate 10 constant rate UDP traffic at 6 Mbps each from a source-destination pair and measure the overall end-to-end network throughput. We repeat this experiment for about 50 different realizations, each time choosing 10 different source-destination pairs. The end-to-end throughput values averaged over all realizations are plotted in Figure 12 and tabulated in Table 4. We can observe from the figure and the table that IntAware algorithm performs consistently better (as evident from the confidence intervals) than the LocBal algorithm, as in the first two experiments of experiment set 1. Furthermore, we observe that the improvement achieved using the INTAWARE algorithm is much higher than those achieved in the single hop experiments. In particular, the maximum improvement achieved is 32.18% (corresponding to 8 channels). This is because, when we have multihop flows, the flows experience interference at every hop between the source and the destination. A channel allocation algorithm that is agnostic of the adjacent channel interference effects, such as LOCBAL will perform poorly in this scenarios. However, the INTAWARE channel allocation algorithm can dynamically change the channel allocation at every hop reducing the overall interference experienced by a traffic flow, resulting in a better end-to-end throughput. This suggests that a interference-aware algorithm is more suitable for a multihop traffic scenario. Again, as in the earlier experiments, we observe from the throughput values that the improvement from INTAWARE decreases as the number of channels available for allocation increases, though the reduction is not substantial.

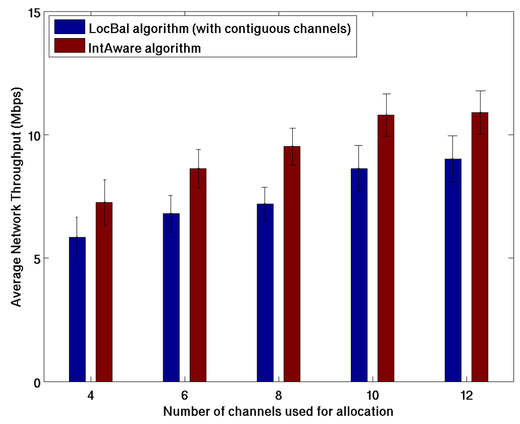

Figure 12. Comparison between the LOCBAL and INTAWARE algorithms for random multihop traffic.

6. RELATED WORK

Intelligent channel allocation is often viewed as a method for mitigating the interference effects in multichannel, multi-radio wireless networks. However, most of the channel allocation algorithms proposed in the literature address only the cochannel interference effects, and therefore often ignore the adjacent channel interference effects. For instance, in [23], the authors have proposed a static channel allocation scheme for multi-radio wireless mesh networks where the objective is to minimize the average size and maximum size of the cochannel interference set. Another work [24], proposes a minimum interference channel assignment in multi-radio networks where the goal is to minimize the number of neighboring nodes operating on any single channel subject to interface constraints. Both the above problems are formulated as NP-hard max-k-cut problems and the authors have provided heuristics for channel allocation. One of the pioneering work in the centralized channel allocation approaches, [25], proposes a neighbor partitioning-based algorithm and a load-aware algorithm for allocating channels in multichannel networks. The goal of the neighbor-partitioning algorithm is to reduce the interference in the network, while the load-aware algorithm is formulated as a network-flow problem.

The channel allocation proposed in [26] and [27], addresses the problem of partial overlap between the channels. However, instead of minimizing the interference effects, the authors propose to use the channel overlaps to improve network connectivity. An interference-aware channel allocation algorithm is proposed in [28] in which the routers switch to a default channel whenever the current channel is perceived to be poor. However, the interference metric used considers only the cochannel interference effects. In [29], the authors have extended their work in [12] and have quantified various forms of radio interference in multi-radio networks. They then compare three different forms of channel allocations, namely ad-hoc, TDM, and FDM. A genetic algorithm-based channel allocation algorithm is proposed in [30]. Bertossi, et al, have proposed minimum number coloring algorithms, such as L(2,1) and L(2,1,1) for various model graphs, where the goal is to find a minimum set of colors for performing a conflict-free coloring of a graph. Within the context of this paper, our goal instead is to find a minimum-conflict coloring given a fixed set of colors. All the works discussed so far are centralized algorithms, which may be easy to implement when there is a centralized server. In ad-hoc and mesh networks, however, a centralized server may not be available and therefore, distributed channel allocation algorithms are required.

Raniwala and Chiueh, have proposed a tree-based distributed channel allocation protocol, where the allocation is based on the total number of links using a channel within the interference range of a node, and the aggregate traffic load on a channel within the interference range [31]. A fully distributed channel allocation protocol for multi-radio mesh networks is proposed in [32] where the objective of this paper was to maximize the utilization of the wireless spectrum over a large network subject to minimizing the cochannel interference over the neighborhood of a node. A randomized distributed algorithm is proposed in [33]. However, the authors have used a delay metric to evaluate their channel allocation. A more relevant channel allocation algorithm for our work are those proposed in [22] and [21]. In [22], the authors propose a distributed channel allocation algorithm in which the hosts make use of the feedback present in the 802.11 networks for estimating the interference. However, unlike the case in our algorithm, a wireless node relies on the traffic information from other nodes, which either may not be always possible in large networks or may not be reliable due to problems such as hidden terminals. In [21], the authors discuss a channel allocation algorithm similar to the naive form of our local-balancing channel allocation algorithm. However, the channel allocation only minimizes the cochannel interference effects in the network. Joint channel allocation and routing algorithms for multichannel mesh networks are proposed in [34, 35], and [36]. The algorithm proposed in [34] is centralized, while [35] and [36] propose a distributed channel allocation and routing algorithm. We do not concentrate on routing algorithms in our work.

7. CONCLUSION

In this work, we have empirically motivated a channel allocation algorithm for a multichannel, multi-radio wireless network that incorporates considerations to the adjacent channel interference effects. We first showed through experiments that interference in a wireless network is mainly due to simultaneous transmissions on the same or adjacent channels, both within a node and from neighboring nodes. We then discussed a simple local balancing channel allocation algorithm, LOCBAL that can reduce the cochannel interference effects by balancing the number of channels used within a two hop neighborhood. We showed that a naive form of this algorithm that uses only a subset of non-contiguous channels (in particular, five out of twelve channels) does not improve the overall performance in the network as this will reduce the total number of simultaneous transmissions in the network (spatial reuse).

Table 4. Throughput (in Mbps) comparison between the LocBal and IntAware algorithms for random multihop traffic.

# Contiguous Channels	LocBal Algorithm (Mbps)	IntAware Algorithm (Mbps)	% Improvement IntAware over LocBal
4	5.85	7.24	23.89%
6	6.80	8.62	26.86%
8	7.2	9.52	32.18%
10	8.64	10.80	25.07%
12	9.02	10.90	20.80%

We then showed that by using more contiguous channels, a better throughput can be achieved compared to the naive approach. We also argued that a further improvement in throughput is possible when additional intelligence by means of interference awareness is built into the algorithm. This motivated our interference-aware algorithm IntAware, which makes use of the information on the current transmission channel from the wireless drivers to perform an interference-free channel allocation. We showed through experiments on a real testbed that IntAware achieves up to 32% improvement over the LocBal algorithm in a real multihop network scenario. Furthermore, we also showed that IntAware performs better than LocBal for single-hop traffic scenarios. We also demonstrated that by carefully selecting the channels, even the interference due to a transmission from a neighboring node can be avoided. However, we wish to explore improved channel allocation strategies for reducing interference from neighboring nodes as a future work.

8. REFERENCES

[1] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 2, pp. 338–404, March 2000.

[2] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*, IEEE 802.11 Working Group Std., September 1999 - 2005.

[3] A. Raniwala and T. c. Chiueh, "Architecture and algorithms for an IEEE 802.11-based multichannel wireless mesh network," in *IEEE Infocom Conference*, 2005.

[4] C. Chereddi, P. Kyasanur, and N. Vaidya, "Design and implementation of a multi-channel multi-interface network," in *ACM REALMAN Workshop*, vol. 5, May 2006, pp. 23–30.

[5] P. Kyasanur, C. Chereddi, and N. Vaidya. (2006, August) Net-x: System extensions for supporting multiple channels, multiple interfaces, and other interface capabilities. [Online]. Available: http://www.crhc.uiuc.edu/wireless/groupPubs.html.

[6] R. Maheshwari, H. Gupta, and S. Das, "Multichannel mac protocols for wireless networks," in *IEEE SECON*, September 2006.

[7] A. Sharma and E. Belding, "FreeMac: Framework for multi-channel MAC development on 802.11 hardware," in *PRESTO Workshop*, August 2008.

[8] P. Bahl, A. Adya, J. Padhye, and A. Wolman, "Reconsidering wireless systems with multiple radios," *ACM SIGCOMM*, October 2004.

[9] K. Ramachandran, I. Sheriff, E. Belding, and K. Almeroth, "A multi-radio 802.11 mesh network architecture," *MONET journal*, vol. 13, pp. 132–146, 2008.

[10] P. Kyasanur and N. Vaidya, "Capacity of multichannel wireless networks: Impact of number of channels and interfaces," in *ACM Mobicom Conference*, vol. 1, August 2005, pp. 43–57.

[11] J. Robinson, K. Papagiannaki, C. Diot, X. Guo, and L. Krishnamurthy, "Experimenting with a multi-radio mesh networking testbed," in *WiNMee Workshop*, April 2005, pp. 1–6.

[12] C.-M. Cheng, P.-H. Hsiao, H. Kung, and D. Vlah, "Adjacent channel interference in dual-radio 802.11a nodes and its impact on multi-hop routing," in *IEEE Globecom Conference*, vol. 1, November 2006, pp. 1–6.

[13] Y. Liu, Y. Xiong, Y. Yang, P. Xu, and Q. Zhang, "An experimental study on multi-channel multi-radio multi-hop wireless networks," in *IEEE Globecom Conference*, vol. 1, November 2005, pp. 5–10.

[14] P. Clifford and D. Leith, "Channel dependent interference and decentralized colouring," *NET-COOP*, 2007.

[15] A. Mishra, V. Shrivastava, S. Banerjee, and W. Arbaugh, "Partially overlapped channels not considered harmful," in *ACM SIGMetrics/Performance Conference*, vol. 34, June 2006, pp. 63–74.

[16] V. Angelakis, A. Traganitis, and V. Siris, "Adjacent channel interference in a multi-radio wireless mesh node with 802.11a/g interfaces," in *IEEE Infocom Conference*, April 2007, Student Poster Abstract.

[17] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou, "A multiradio unification protocol for IEEE 802.11 wireless networks," in *Broadnets Conference*, vol. 1, October 2004, pp. 344–354.

[18] A. Mishra, E. Rozner, S. Banerjee, and W. Arbaugh, "Exploiting partially overlapped channels in wireless networks: Turning a peril into an advantage," in *Internet Measurement Conference*, October 2005.

[19] V. Navda, A. Kashyap, and S. Das, "Design and evaluation of imesh: an infrastructure-mode wireless mesh network," in *IEEE WoWMoM Symposium*, June 2005.

[20] K. Ramachandran, K. Almeroth, and E.M.B.-Royer, "A novel framework for the management of large-scale wireless network testbeds," in *WinMee Workshop*, April 2005.

[21] P. Kyasanur and N. Vaidya, "Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks," *ACM SIGMOBILE MC2R*, vol. 10, pp. 31–43, January 2006.

[22] D. Leith and P. Clifford, "A self-managed distributed channel selection algorithm for WLANs," in *RAWNET*, 2006.

[23] A. Das, R. Vijayakumar, and S. Roy, "Static channel assignment in multi-radio multi-channel 802.11 wireless mesh networks: Issues, metrics and algorithms," in *IEEE Globecom Conference*, vol. 1, November 2006, pp. 1–6.

[24] A. Subramanian, H. Gupta, and S. Das, "Minimum

interference channel assignment in multi-radio wireless mesh networks," in *IEEE SECON Conference*, vol. 1, June 2007, pp. 481–490.

[25] A. Raniwala, K. Gopalan, and T.-C. Chiueh, "Centralized channel allocation and routing algorithms for multi-channel wireless mesh networks," *ACM MC2R*, vol. 8, pp. 50–65, April 2004.

[26] A. Mishra, S. Banerjee, and W. Arbaugh, "Weighted coloring based channel assignment for wlans," *ACM SIGMOBILE MC2R*, vol. 9, pp. 19–31, July 2005.

[27] A. Mishra, V. Brik, S. Banerjee, A. Srinivasan, and W. Arbaugh, "A client-driven approach for channel management in wireless lans," in *IEEE Infocom Conference*, vol. 1, April 2006, pp. 1–12.

[28] K. Ramachandran, E. Belding, K. Almeroth, and M. Buddhikot, "Interference-aware channel assignment in multi-radio wireless mesh networks," in *IEEE Infocom*, vol. 1, April 2006, pp. 1–9.

[29] C.-M. Cheng, P.-H. Hsiao, H. Kung, and D. Vlah, "Parallel use of multiple channels in multi-hop 802.11 wireless networks," in *IEEE Milcom Conference*, vol. 1, September 2006, pp. 1–9.

[30] H. Liu, H. Yu, X. Liu, C.-N. Chuah, and P. Mohapatra, "Scheduling multiple partially overlapped channels in wireless mesh networks," in *IEEE ICC Conference*, vol. 1, June 2007, pp. 3817–3822.

[31] A. Raniwala and T.-C. Chiueh, "Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network," in *IEEE Infocom Conference*, vol. 3, March 2005, pp. 2223–2234.

[32] B.-J. Ko, V. Misra, J. Padhye, and D. Rubenstein, "Distributed channel assignment in multi-radio 802.11 mesh networks," in *IEEE WCNC Conference*, vol. 1, March 2007, pp. 3978–3983.

[33] M. Shin, S. Lee, and Y.-A. Kim, "Distributed channel assignment for multi-radio wireless networks," in *IEEE MASS Conference*, vol. 1, October 2006, pp. 417–426.

[34] X. Meng, K. Tan, and Q. Zhang, "Joint routing and channel assignment in multi-radio wireless mesh networks," in *IEEE ICC Conference*, vol. 8, June 2006, pp. 3596–3601.

[35] 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 JSAC*, vol. 24, pp. 1972–1983, November 2006.

[36] A. Mohsenian and V. Wong, "Joint optimal channel assignment and congestion control for multi-channel wireless mesh networks," in *IEEE ICC Conference*, vol. 5, June 2006, pp. 1984–1989.