Comparison of TDMA and CSMA MAC performance in Sub-GHz band

R&D Project Report

Gaurang Naik (123079009)
Sudesh Singhal (123079004)

Department of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai – 400076

Contents

Chapter 1

Introduction

Spectrum scarcity is being faced all across the globe and thus spectrum has become a very expensive and valuable resource. Facing this spectrum crunch, researchers all around are focusing on new technologies and techniques to efficiently use the existing frequency bands. One research area that has emerged as a result of these activities is the use of what are known as 'TV White Spaces' for secondary networks. Terrestrial TV Services operate in the Very High Frequency (VHF) and Ultra High Frequency (UHF) bands all across the world. TV White Spaces (TVWS) are portions of the spectrum in these TV bands which are not being used at a given point in time or location by the TV transmitters (primary users). Potentially, these portions of the spectrum can be utilized by other users on a secondary basis, without causing harmful interference to the primary users. Such users are termed Secondary users in the literature.

Several studies have been carried out in different regions of the world to quantify the available TV White Spaces [1-4]. It has been observed that in all regions of the world, portions of the TV Spectrum are under-utilized or unutilized. Particularly, in India the 470-585 MHz band is almost completely unutilized [5]. As a result of such observations, several use cases have been proposed for the use of TV White Spaces. One such use case is the use of TVWS for provisioning rural broadband using multi-hop wireless mesh networks. The overview of such a mesh network is as shown in Figure 1.

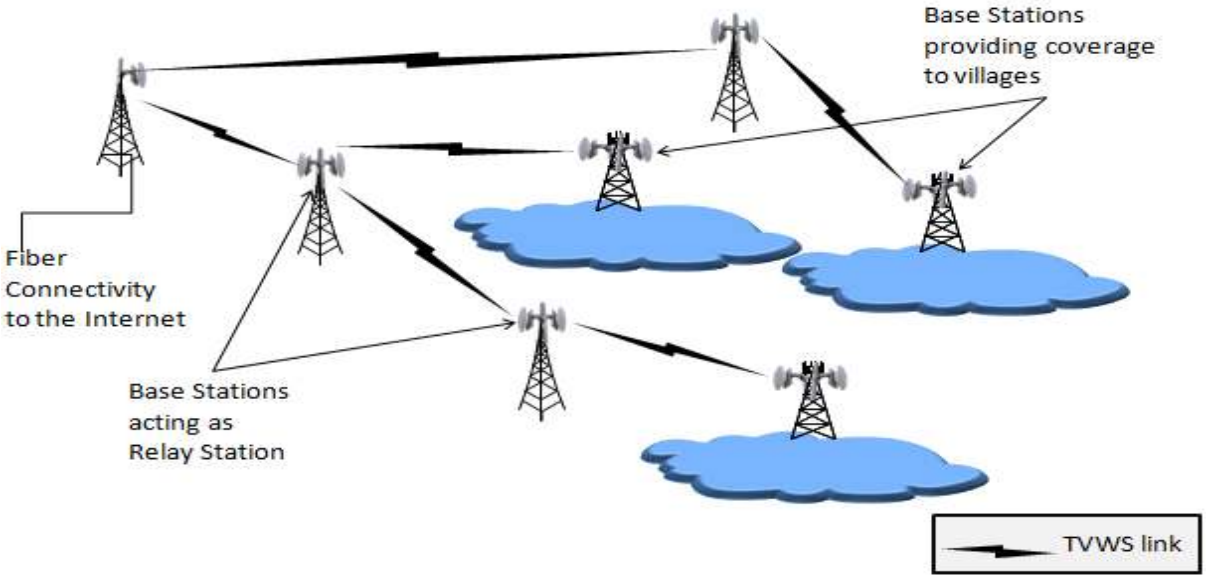

Figure 1: Proposed rural broadband wireless backhaul network is depicted. The blue clouds represent a village which is under coverage by the TV white space base station. The intermediate base stations act as relays. All the relays terminate at a base station with Fiber connectivity.

In the past, such wireless mesh networks have been deployed using the 2.4GHz (802.11b/g) or 5GHz (802.11a) WiFi like networks [6]. However, the 2.4GHz and 5GHz band suffer from poor propagation characteristics as compared to the sub-GHz frequency bands. Moreover, the 2.4GHz band is crowded due the use of WiFi in offices and homes in addition to the use of technologies like Bluetooth, ZigBee

etc. This interference is less due to the fact that WiFi nodes used for the deployment of such mesh network operate at larger heights from the ground (10-30m depending on the cost of network permissible) as compared to the other operating devices. However, the sub-GHz band is practically free from all interference. Hence, the use of TV White Space for the deployment of such a mesh network is expected to yield better performance as compared to the 2.4GHz and 5GHz WiFi network.

While the deployment of such wireless mesh networks is possible using several PHY and MAC layer technologies, the traditional CSMA MAC protocol used for WiFi and the TDMA based MAC have been used in practical site deployments [7]. In this work, we compare the performance of the TDMA based LiTMAC protocol and the CSMA MAC in a 4-node network within the IIT Bombay campus in the sub-GHz band. We have taken throughput measurements for two different topologies, point-to-multipoint and multi-hop, using the TDMA based MAC; while we measured the throughputs for the CSMA based network in the point-to-multipoint topology by the configuring the nodes in the Access Point (AP) – Client (STA) mode.

The rest of this document is organized as follows. Chapter 2 describes the setup of the hardware used in our deployment. Details of the topology setup and experiments conducted are mentioned in chapter 3. We present and discuss the results of our measurements in Chapter 4. In chapter 5, we mention some of the problems we faced during the course of our work and the solutions we could find. Finally chapter 6 provides some concluding remarks.

Chapter 2

Hardware Setup

The US Federal Communications Commission (FCC) permitted the use of TVWS for secondary usage in November 2008 and since them several manufacturers have released radio devices that operate in the TV bands. Most of these devices use proprietary PHY and MAC layer techniques and are generally very expensive (~$7000-$10000 for a kit comprising of 1 Base Station and 1-2 clients). This is primarily because of the stringent out-of-band emission constraints and database query and/or spectrum capabilities imposed by the FCC and other regulatory bodies across the world. However, for preliminary experiments, such stringent requirements need not be imposed and low-cost WiFi-like RF cards are available off the shelf.

We have used RF cards (DL-535 TVWS card) manufactured by the US based manufacturer Doodle Labs [8]. These cards are simple modifications of the IEEE 802.11g cards which can operate with the same CSMA based MAC used for 2.4 GHz WiFi. The radio front-end of the card is modified to operate in the sub-GHz band instead of the standard 2.4GHz band. The Channel 1 of DL-535 card translates to 540 MHz center frequency, while channel 11 translates to 590 MHz center frequency. These cards can be operated using ath5k and madwifi drivers. The ath9k drivers do not function in these cards.

We used a combination of Routerboard RB 411AR and RB 433AH boards as the baseband processors in the deployment of the network. The throughput measurements for the TDMA based LiTMAC were performed using the OpenWrt Backfire 10.03 OS with madwifi drivers, while those for CSMA network were performed using OpenWrt Attitude Adjustment 12.09 OS using ath5k drivers.

In this chapter, we outline in brief the steps used in configuring the RouterBoard 411AR and 433AH boards.

CSMA Configuration

We first configured 4 boards using the Attitude Adjustment OS to operate using the standard CSMA MAC protocol. The process for completing the setup of each node can be broken down into the following steps.

1. Make kernel and rootfs files for the desired OS
2. Flash these files into the RouterBoards
3. Configure relevant files and install selected tools

Make kernel and rootfs files for the desired OS

- Go to the directory where you wish to download the source code
- Download Attitude Adjustment 12.09 source using the following command

```
svn co svn://svn.openwrt.org/openwrt/branches/attitude_adjustment
```

- Untar the file and copy it in home folder
- Go to the openwrt folder

```
cd openwrt
```

- We now need to configure the OS to include the relevant tools and drivers. This can be done using the make menuconfig command

```
make menuconfig
```

- This will open a User Interface. (If it does not, there might be some dependencies missing. Install them and try the command again). In order to install the Operating System in the NAND Memory of the Routerboard, we first need to get the board running using network boot. To this, we must prepare the ramdisk image file.

- Select the following components:

```
Target System: AR71XXX
Subtarget: Devices with NAND flash (mostly Mikrotik)
Target Profile: Atheros WiFi (ath5k)
Target Images: ramdisk
Kernel Modules -> Wireless Drivers: kmod-ath5k (Make sure it is
selected as '*' and not 'm')
Network: wpa-supplicant
Base System: wireless-tools
```

- Once this is completed, compile the OS using the command

```
make V=99
```

- If there are no errors in the compilation, the ramdisk image file will be formed in the following directory

```
~/openwrt/bin/ar71xx
```

Rename this ramdisk file as 'vmlinux'.

Flash kernel and rootfs files into the RouterBoard

- Using this image, the RouterBoard will boot from the network. However, for field deployments, we want the OS to be installed permanently into the NAND Memory of the RouterBoard. To do this, we first need to create the kernel image file and the rootfs file. This can be done using follows.

```
make menuconfig
Keep all other configurations the same as before and change the
Target Images
Target Images: tar.gz, jffs2, squashfs
Exit the configuration screen and compile the OS using the command:
make V=99
```

- The ramdisk file has to be first loaded onto the RouterBoard using network boot. To do this, we first need to create an HTTP and DHCP server in the local host system. This is done as follows.

```
Install and Configure DHCP & TFTP Server
sudo apt-get install atftpd
sudo apt-get install dhcp3-server
sudo mkdir /tftpboot
Go to the directory ~/openwrt/bin/ar71xx
sudo cp vmlinux /tftpboot/vmlinux
```

```
sudo atftpd --bind-address <ip-addr of the local host system> --
logfile ~/log  --daemon
Configure file /etc/dhcp/dhcpd.conf file appropriately, and then
start dhcpd via the command
sudo dhcpd -cf /etc/dhcp/dhcpd.conf --no-pid  eth0
```

- Now, switch on the RouterBoard, and enable network boot. Connect the RouterBoard to the local host system using serial cable. To gain access to the system, minicom needs to be installed in the local host system. Install minicom using the following command.

```
sudo apt-get install minicom
```

Now, to run minicom, type the following.

```
sudo minicom
```

- If the above configurations are correctly done, the Routerboard will transfer the ramdisk file into its RAM and boot using network boot. Once the device boots, execute the following.

```
ifconfig br-lan down
brctl delbr br-lan
ifconfig eth0 10.107.48.50
passwd
<Set Password>
```

- Now, in a new terminal go the directory containing the kernel image file and the rootfs file

```
cd ~/openwrt/bin/ar71xx/
The kernel file (openwrt-ar71xx-nand-vmlinux.elf) and the rootfs file
(openwrt-ar71xx-nand-rootfs.tar.gz) need to be copied into the /tmp
directory of the RouterBoard.
scp openwrt-ar71xx-nand-rootfs.tar.gz -ar71xx-nand-vmlinux.elf
root@10.107.48.50:/tmp/
```

- At this point, the RouterBoard has the kernel image and rootfs file in its /tmp/ directory. To install OpenWrt on the RouterBoard run following set of commands.

```
First mount the kernel image in the mtdblock1 partition
mount /dev/mtdblock1 /mnt/
mv /tmp/ openwrt-ar71xx-nand-vmlinux.elf /mnt/kernel
umount /mnt

Now, mount the rootfs (utilities) in the mtdblock2 partition
mount /dev/mtdblock2 /mnt/
cd /mnt/
tar -xzvf /tmp/ openwrt-ar71xx-nand-rootfs.tar.gz
cd ..
umount /mnt/

sync
```

- Both, the kernel image and the utilities have been mounted on the RouterBoard. Reboot the system now.

```
reboot
```

- When the system reboots, make sure you change the boot option from netboot to Boot from NAND. If this is not done, then all the steps described above (From `ifconfig br-lan down` onwards need to be redone.)

- If the process described above has been done correctly, then the RouterBoard will boot without any errors. We now need to configure the boards to communicate with each other and enable performing throughput measurements using the boards.

Configure relevant files and install selected tools

- Once the board reboots, we first need to set the network parameters in the following files

  `/etc/config/network and /etc/config/wireless`

  - **In /etc/config/network**

We change the 'lan' interface as follows.

```
config interface 'lan'
       option ifname 'eth0'
       option proto 'static'
       option ipaddr '10.107.48.10*'
       option netmask '255.255.240.0'
       option gateway '10.107.63.250'
```

We then add a new network interface for the radio interface as follows.

```
config interface 'wlan'
       option ifname 'wlan0'
       option proto 'static'
       option ipaddr '192.168.1.*'
       option netmask '255.255.255.0'
```

The value * varied from 1 through 4 for the four nodes in our network.

  - **In /etc/config/network**

We first need to enable the wireless interface. Delete or comment the following line.

```
option disabled 1
```

Make sure you comment the above line corresponding to the correct wireless interface. This problem will not arise in case of RB433AH boards. However, in case of RB411AR, the board has an internal 2.4GHz WiFi card. The first wireless interface in the file corresponds to this internal WiFi card and need not be enabled as we are not using this band for our measurements.

A sample configuration of the wireless interface is as follows.

```
config wifi-device radio0
       option type mac80211
       option channel 7
       option hwmode 11g
       option path 'pci0000:00/0000:00:14.0'

config wifi-iface
       option device radio0
       option network wlan
```

```
option mode sta
option ssid OpenWrt
option encryption none
```

Note:

- ✓ The option network must contain the name of the network interface which was created for the wireless interface in the /etc/config/network file above.
- ✓ Make sure all nodes have the same ssid and channel numbers. Also, if the nodes are to operate in AP-Client mode, then one of the nodes is to be configured in the AP mode and the remaining in client mode. To configure a node in the AP mode, simply replace the 'option mode sta' by 'option mode ap'. All other configuration values remain the same.

- At this stage, the basic configuration of the nodes is done. Reboot the system, connect the antenna to the card and check if the nodes connect to each other.

- If the nodes are configured in the AP-Client mode, then all clients associate to the AP. This can be seen in the kernel logs, which can be viewed using

```
dmesg
```

It can also be verified by using the iwconfig command

```
iwconfig
```

A sample output of the command at a client is as shown below.

```
wlan0     IEEE 802.11bg  ESSID:"OpenWrt"
          Mode:Managed  Frequency:2.462 GHz Access Point: 00:30:1A:46:09:5B
          Bit Rate=24 Mb/s   Tx-Power=27 dBm
          RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=61/70  Signal level=-49 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:98  Invalid misc:951   Missed beacon:0
```

Since the client is associated to the Access Point, we see the MAC address of the Access Point. Note that the Frequency shown by this command is 2.462GHz, which is the frequency corresponding to channel 7 for a normal WiFi card. However, the card operates at 570MHz.

- If all clients are associated to the AP, then we can start communication between the clients and the AP. Simply ping any of the node in the network.

```
ping 192.168.1.1
```

If the above command returns an output, the nodes are connected to the AP and can communicate with one another.

- To measure the TCP and UDP throughputs, we need to install a tool like iperf at every node. This can be done as follows.
  - Connect the RouterBoard to the Internet using a switch.
  - Gain access to the RouterBoard using serial cable or remote login (`ssh`).
  - Installation of any external package is done using `opkg`.
  - First update the list of packages using
    `opkg update`

    - Now search for iperf in the list of packages
      `opkg list | grep iperf`
    - Install operf using
      `opkg install iperf`
- If the nodes communicate with each other (ping each other), then we can run iperf server at any node and measure the throughput of any link by running an iperf client at the desired node and connecting the the iperf server.

- We now have four nodes ready for deployment which use the standard CSMA MAC using the OpenWrt Attitude Adjustment 12.09 OS.

LiTMAC Configuration

We then configured 4 boards using the Backfire OS to operate using the TDMA based LiTMAC. While most of steps in this configuration remain the same as above, a few TDMA specific configurations are needed to be done. The steps for installing Backfire 10.03 OS with LiTMAC are outlined below.

Make kernel and rootfs files for the desired OS

- Go to the directory where you wish to download the source code.
- Download Backfire 10.03 source from the following link

```
http://downloads.openwrt.org/backfire/10.03/backfire_10.03_source.tar.bz2
```

- Untar the file and copy it in home folder.
- Run the command make kernel_menuconfig from the folder. Select the following option for Kernel type

```
Kernel type -> High Resolution Timer Support -> Timer frequency 1000 HZ
```

- Download all contents from the folder `www.ee.iitb.ac.in/student/~gaurangnaik/projectwork/fractel`.
- Paste the Makefile in `~/backfire_10.03/package/madwifi`
- Delete the contents of `~/backfire_10.03/package/madwifi/patches`
- Paste the folders `fractel_readlog/` and `fractel_routing_daemon/` in `~/backfire_10.03/package`
- Paste `madwifi-trunk-r3314.tar.gz` in `~/backfire_10.03/dl/`
- Go to the openwrt folder

  ```
  cd backfire_10.03
  ```

- Run the make menuconfig command

  ```
  make menuconfig
  ```

- Note that the same ramdisk file use during Attitude Adjustment installation can be used for the installation of Backfire.

  ```
  Target: AR71XXX
  Default: no WiFi
  Generate tar.gz, squashfs
  Wireless Modules: ath5k (Make sure it is selected as '*' and not 'm')
  wpa-supplicant
  ```

```
Utilities-> Check on fractel_readlog and fractel_routing_daemon
```

Compile the OS using the following command.

```
make V=99
```

- Load the ramdisk file into the RouterBoard in the same way as described during Attitude Adjustment installation.

Flash kernel and rootfs files into the RouterBoard

- Now, switch on the RouterBoard, and enable network boot. Connect the RouterBoard to the local host system using serial cable. To run minicom, type the following.

```
sudo minicom
```

- If the above configurations are correctly done, the Routerboard will transfer the ramdisk file into its RAM and boot using network boot. Once the device boots, execute the following.

```
ifconfig br-lan down
brctl delbr br-lan
ifconfig eth0 10.107.48.50
passwd
<Set Password>
```

- Now, in a new terminal go the directory containing the kernel image file and the rootfs file

```
cd ~/openwrt/bin/ar71xx/

The kernel file (openwrt-ar71xx-vmlinux.elf) and the rootfs file
(openwrt-ar71xx-rootfs.tgz) need to be copied into the /tmp directory
of the RouterBoard.

scp openwrt-ar71xx-vmlinux.elf openwrt-ar71xx-rootfs.tgz
root@10.107.48.50:/tmp/
```

- At this point, the RouterBoard has the kernel image and rootfs file in its /tmp/ directory. To install OpenWrt on the RouterBoard run following set of commands.

```
First mount the kernel image in the mtdblock1 partition
mount /dev/mtdblock1 /mnt/
mv /tmp/ openwrt-ar71xx-vmlinux.elf /mnt/kernel
umount /mnt

Now, mount the rootfs (utilities) in the mtdblock2 partition
mount /dev/mtdblock2 /mnt/
cd /mnt/
tar -xzvf /tmp/ openwrt-ar71xx-rootfs.tgz
cd ..
umount /mnt/

sync
```

- Both, the kernel image and the utilities have been mounted on the RouterBoard. Reboot the system now.

```
reboot
```

- Again, when the system reboots, make sure you change the boot option from netboot to Boot from NAND. If this is not done, then all the steps described above (From `ifconfig br-lan down` onwards need to be redone.)

- If the process described above has been done correctly, then the RouterBoard will boot without any errors. We now need to configure the boards to communicate with each other and enable performing throughput measurements using the boards.

Configure relevant files and install selected tools

- Once the board reboots, the network configuration using the `/etc/config/network` and `/etc/config/wireless` files must be done in the same way as in case of Attitude Adjustment.

- In case of LiTMAC, several TDMA specific parameters need to be configured manually. In this case many of the parameters set in `/etc/config/network` and `/etc/config/wireless` files are overridden.

- In case of LiTMAC, one node in the network is responsible for performing Time Synchronization of all other nodes. All other nodes synchronize their time slots with respect to this one node, called the root node. All other nodes in the system connect to each other using a tree topology based on the signal strength. The first node, if in communication range of the root node, connects to the root node. The second node, depending upon whether it receives better signal strength from the root node or the non-root node connects to the respective node. This process continues to result in a tree topology.

- Majority of the TDMA specific configurations are done using the files `config.cfg` and `startup.sh`. The `config.cfg` file contains values of TDMA specific parameters, while the `startup.sh` file makes all necessary changes at the startup. These files can be placed anywhere in the system. We place it in the `/root` directory of the RouterBoard.

- The configuration files for the root node in our network are as follows.

`config.cfg`

```
interval=5
node_id=0
self_node_ip=192.168.1.1
no_of_control_slots=1
no_of_contention_slots=3
no_data_slots=99
max_no_of_rssi_values=8
rssi_sense_timeout=-1
root_ip=192.168.1.1
fractel_data_rate=11
frame_length=4
slot1=0

interval -> specifies the duration of one time slot in msec
node_id -> specifies the id of the node. This is 0 for the root node
and non-zero for non-root node.
self_node_ip -> specifies the IP address of the current node
no_of_control_slots -> specifies number of control slots
no_of_contention_slots -> specifies number of contention slots
no_data_slots -> specifies number of data slots
root_ip -> specifies the IP address of the root node
```

```
fractel_data_rate -> specifies the data rate of the card
```

Note: LiTMAC does not support Link Rate Adaptation. The value of data rate set in config.cfg file and startup.sh file remains constant unless changed manually.

```
frame_length -> this is equal to the number of nodes in the network
slot1 -> this varies from 0 to (frame_length-1).
slot1=0 for the root node, for other nodes the values vary from 1 to
(frame_length-1).
```

startup.sh

```
wlanconfig ath0 destroy
wlanconfig ath0 create wlandev wifi1 wlanmode monitor
sleep 5
ath=`ifconfig -a| grep "ath" | cut -f 1 -d ' '`
iwconfig $ath channel 7
iwconfig $ath essid meas
iwconfig $ath txpower 23dbm
iwconfig $ath rate 6M fixed
ifconfig $ath down
ifconfig $ath 192.168.1.1 netmask 255.255.0.0 up
cat /root/config.cfg > /proc/net/madwifi/ath0/fractel_config
echo 0 > /proc/sys/dev/wifi1/diversity
echo 1 > /proc/sys/dev/wifi1/rxantenna
echo 1 > /proc/sys/dev/wifi1/txantenna
sysctl -w dev.wifi1.disable_cca=7
sysctl -w dev.wifi1.acktimeout=30
sysctl -w dev.wifi1.intmit=0
sysctl -w dev.wifi1.ofdm_weak_det=1
sysctl -w dev.wifi1.noise_immunity=4
rm /root/outfile
fractel_readlog /root/outfile &
fractel_routing_daemon /root/routing_log &
iptables –P FORWARD ACCEPT
iptables –F FORWARD
```

- The `startup.sh` file runs when the device boots up. It is used for setting parameters like channel number, ssid, transmit power, rate and the IP address. It also runs the fractal routing daemon which generates routes based on SNR values. The `iptables` commands at the end of the file are used to forward the packets across hops.

- All nodes operate in the monitor mode in LiTMAC. Once the `fractal_routing_daemon` runs on a particular node, it forms the tree topology by connecting to the node with strongest signal. All nodes connect in such a manner and form a network. Once the node reboot, try to ping all nodes from any particular node.

- If the nodes communicate with each other (ping each other), then we can run iperf server at any node and measure the throughput of any link by running an iperf client at the desired node and connecting the the iperf server.

We now have four nodes ready for deployment which uses the TDMA based LiTMAC using the OpenWrt Backfire 10.03 OS.

Chapter 3

Experimental Setup

The objective of our work is to compare the performance of the TDMA based LiTMAC and the CSMA MAC in various settings. To evaluate the MAC performance, we set up a network within the IIT Bombay campus comprising of four nodes. In each case (LiTMAC as well as CSMA), the frequency of operation of all nodes is 570 MHz. The positioning of the four nodes is as shown in Figure 2. At the H14 and GG node, we use a Yagi antenna with a gain of 11 dBi, while at the MB and VMCC nodes, we use an omni-directional antenna with a gain of 3 dBi. The distances of each pair of nodes is given in Table 1.

Nodes	Distance (km)
H14-GG	1.14
H14-MB	0.98
H14-VMCC	1.19
MB-GG	0.18
MB-VMCC	0.21
GG-VMCC	0.13

The shortest distance is between the nodes GG and VMCC. However, the antenna at GG is directional and points in between H14 and MB. As a result, a direct connection between GG and VMCC nodes is almost never formed.

Figure 2: Positioning of the four nodes within the IIT Bombay campus. The four nodes are placed at the rooftops of Hostel 14 C-wing (H14), VMCC, GG Building (GG) and the Main Building (MB).

We first carry out the throughput measurements using LiTMAC. The node at Hostel 14 (H14) was configured as the root node, while the other three nodes are non-root nodes. Experiments were carried out in two different topologies, (i) point-to-multipoint and (ii) multi-hop.

In the point-to-multipoint topology (Figure 3), all the non-root nodes (GG, MB and VMCC) connect directly to the root node (H14). The route to every other non-root node is via the root node. A sample output of the `route` command at the GG node in this topology is as below.

```
route

Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
Iface
192.168.1.2     192.168.1.1     255.255.255.255 UGH   0      0      0 ath0
192.168.1.4     192.168.1.1     255.255.255.255 UGH   0      0      0 ath0
```

Figure 3: Point to multipoint topology. All non-root nodes (GG,MB and VMCC) connect directly to the root node (H14).

We first run the iperf TCP and UDP server at each of the four nodes. We run iperf TCP client at one node and run 5 TCP sessions of 10 seconds each at each of the remaining three nodes. This is followed by 5 UDP sessions of 10 seconds at each of the remaining three nodes, and then 5 UDP Bidirectional sessions of 10 seconds each. We repeat this process at all the four nodes at a particular data rate. As mentioned in Chapter 2, LiTMAC does not support Link Rate Adaptation. This means that the data rate once set remains constant across the entire duration of the measurements.

We fixed the data rates to 5 different values, 6 Mbps, 9 Mbps, 12 Mbps, 18 Mbps and 24 Mbps and carried out the TCP, UDP and UDP Bidirectional throughput measurements as described above. At data rates higher than 24 Mbps (36 Mbps, 48 Mbps and 54 Mbps) the cards were unstable and lost connectivity frequently and conducting measurements at these data rates was not possible.

Next, we repeated the same experiments, i.e. 5 sessions of 10 seconds each of TCP, UDP and UDP Bidirectional throughput measurements at 5 data rates (6 Mbps, 9 Mbps, 12 Mbps, 18 Mbps and 24 Mbps) pairwise at each of the nodes connected via a multi-hop topology (Figure 4). A sample output of the `route` command at the GG node in this topology is as below.

```
route

Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
Iface
192.168.1.4     192.168.1.1     255.255.255.255 UGH   0      0      0 ath0
```

The output of the `route` command suggests that the MB and the H14 nodes are connected to the GG node directly, while the route to the VMCC node is via H14.

Figure 4: Multi-hop topology. The non-root nodes at GG and VMCC connect directly to the root node (H14, while the node at MB connects to the other non-root nodes via the GG node.

It must be noted that in each of the above settings, we run iperf sessions between only one client-server pair.

We tabulated the results obtained for measurements of the TCP throughput and throughput, jitter and percentage of packets lost in UDP and UDP Bidirectional iperf sessions. We average out the throughput, jitter and the lost packet percentage across reverse links (e.g. MB-H14 and H14-MB). We present the results for both the above topologies in the next Chapter along with the analysis of results.

Once the throughput measurements for the LiTMAC based networks are completed, we replaced the boards by CSMA based nodes. In this case, the topology is similar to the one shown in Figure 3. The node at H14 is configured as the Access Point, while the remaining three nodes, GG, MB and VMCC are configured to operate as clients. As described above, we run iperf TCP client at one node and run 5 TCP sessions of 10 seconds each at each of the remaining three nodes. This is followed by 5 UDP sessions of 10 seconds at each of the remaining three nodes, and then 5 UDP Bidirectional sessions of 10 seconds each. This is done at all four nodes for fixed data rates. We fix the data rates at each node using the following command.

```
iwconfig wlan* rate #M fixed

where, * is either 0 or 1. * is 1 for RB411AR boards, while it is 0 for
RB433AH boards.
# is one of the 5 data rates, 6, 9, 12, 18 or 24.
```

We again tabulate the results for the CSMA based network and present the results in the next chapter.

We also made an effort to form the topology as shown in Figure 4 by configuring all the boards in ad-hoc mode and statically configuring routes at each of the nodes to form the desired topology. However, the ping values observed in this case were extremely high and the throughput observed even at data rate as high as 24Mbps was as low as a 700 kbps. Therefore, we did not proceed with these measurements.

Chapter 4

Results & Analysis

As described in Chapter 3, we measured the TCP, UDP and UDP Bidirectional throughput at all nodes in two different topologies (point-to-multipoint and multi-hop) for the TDMA based LiTMAC. We also measured the TCP, UDP and UDP Bidirectional throughput at all nodes in the AP-Client mode in the standard CSMA topology. We present the results of these measurements in this chapter.

LiTMAC Performance

We first calculate the expected throughput at each node in case of LiTMAC at each data rate. This helps us in deciding whether our experimental setup has all the parameters correctly set. Moreover, it establishes the relationship between the theoretical values and practically observed values. A sample calculation of the expected throughput at 6 Mbps fixed rate and other fractal parameters as seen in the config.cfg file is as shown below.

The following table shows the time taken for the transmission for each portion of packet at 6 Mbps data rate.

Description	Bytes	Time (usec)
UDP Payload	1470	1960
UDP Header	8	10.667
IP Header	20	26.667
Ethernet Header	14	18.667
CRC Tailer	4	5.333
Fractel Data Header	32	42.667
Total	**1548**	**2064**

Now, we have a total of 103 slots (99 data slots + 1 control slot + 3 contention slots). Each slot occupies a duration of 5 msec; and the guard time is 100 microsec. Thus, the total transmission time is 4900 microsec. The total time for 103 slots = 103 * 5 msec = 515 msec. Since there are a total of four nodes in the network, the 99 data slots are divided equally among the four nodes. This gives every node an average of 99/4 = 24.75 slots in 103 slots interval. This translates to 24.75 * 1000 / 515 = 48 slots / sec. Thus, on an average, each node will get 48 transmission slots/second.

As seen in the table, each packet takes 2064 microsec for transmission. Therefore, the total number of packets/slot = floor(4900/2064) = 2. This leads to 48 * 2 = 96 packets/sec. Now, out of the 1548 bytes of data, only 1470 bytes represent useful information. Thus, total data transferred/second = 96 * 1470 * 8 / 10^(6) = **1.12896 Mbits/sec.** Hence, to conclude, at a data rate of 6 Mbps and using the parameters as given in the config.cfg file, the expected throughput is 1.12896 Mbps. Similarly, the expected UDP throughput for data rates 9M, 12M, 18M and 24M have been calculated.

In the point-to-multipoint topology, the H14-GG, H14-MB and H14-VMCC links are single-hop links, while the GG-MB, GG-VMCC and MB-VMCC links are two-hop links. In the multi-hop topology, the H14-GG, H14-VMCC and MB-GG links are single-hop links. The MB-H14 and GG-VMCC are two-hop links, while the MB-VMCC is a three-hop link. For each throughput measurement (TCP, UDP and UDP

Bidirectional) over a link, we have averaged 10 readings (5 in each direction; for e.g. MB-GG and GG-MB).

The following set of graphs show the results obtained in TCP, UDP and UDP Bidirectional throughput measurements at different throughputs and in different links. The graphs also compare the obtained results with the expected UDP throughput.

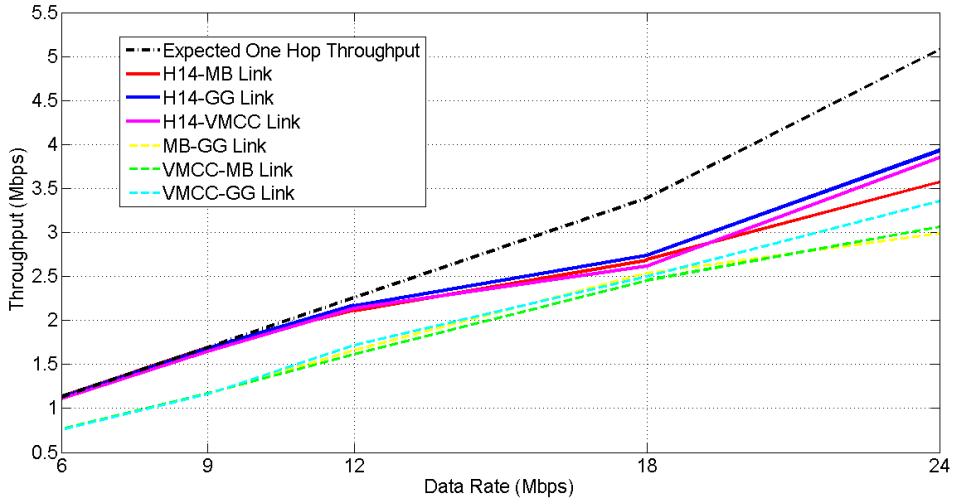

Figure 5a. TCP Throughput for various links in point-to-multipoint topology as shown in Figure 3.

For low data rates, the expected UDP throughput and the observed TCP throughput are almost similar. However, as the data rate increases, the observed TCP throughput does not match up to the expected UDP throughput. In the above and below figures, the solid lines represent single-hop links, while the dotted line represents two-hop links. It is observed that though the one-hop and two-hop throughputs do not vary by a large value, the one-hop throughput lines form one cluster while the two-hop throughput lines form part of another cluster. Somewhat similar trend is followed in the UDP throughputs as well, as shown below.

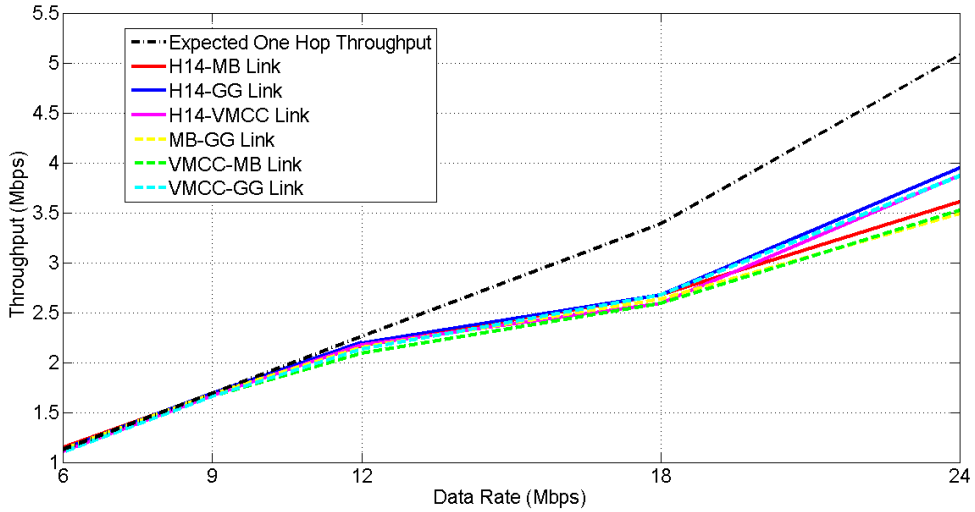

Figure 5b. UDP Throughput for various links in point-to-multipoint topology as shown in Figure 3.

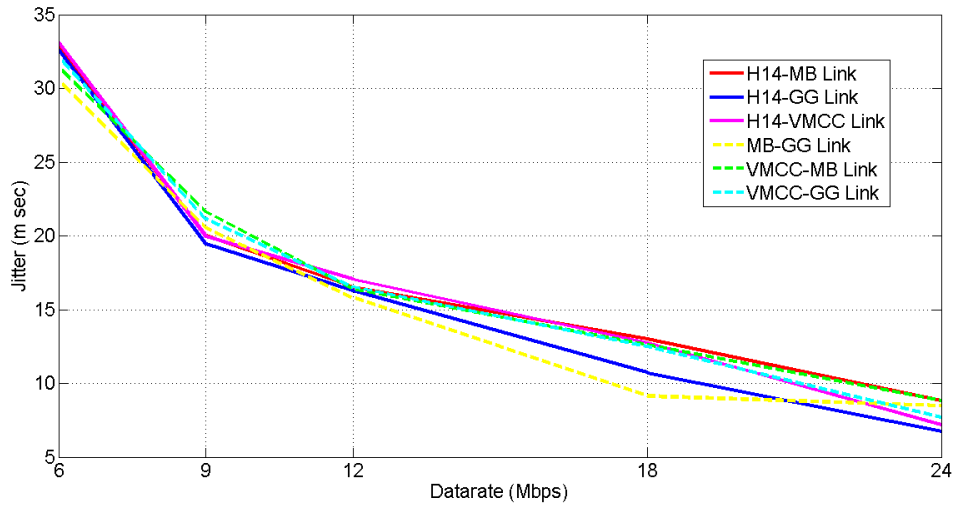

Figure 5c. UDP Jitter values for various links in point-to-multipoint topology as shown in Figure 3.

It is clearly observed from the above graph that as the data rate increases, the jitter decreases.

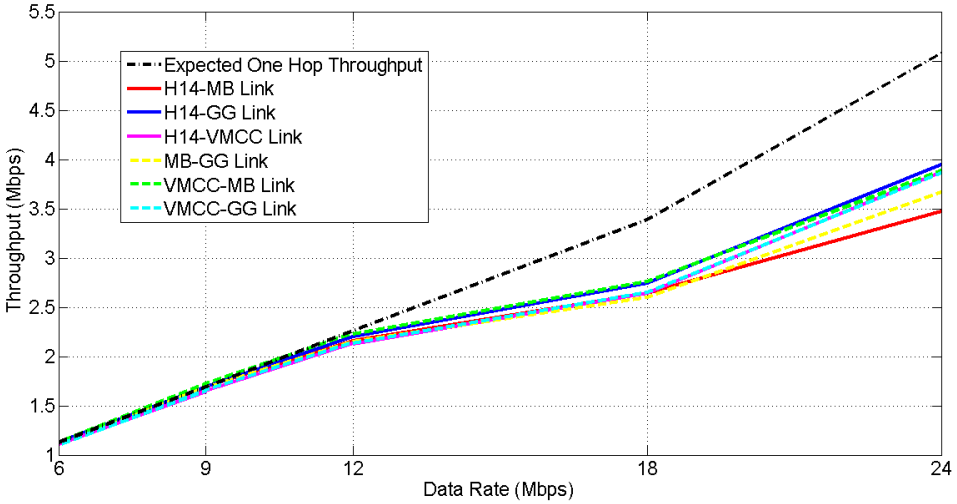

Figure 5d. UDP Bidirectional throughput for various links in point-to-multipoint topology as shown in Figure 3.

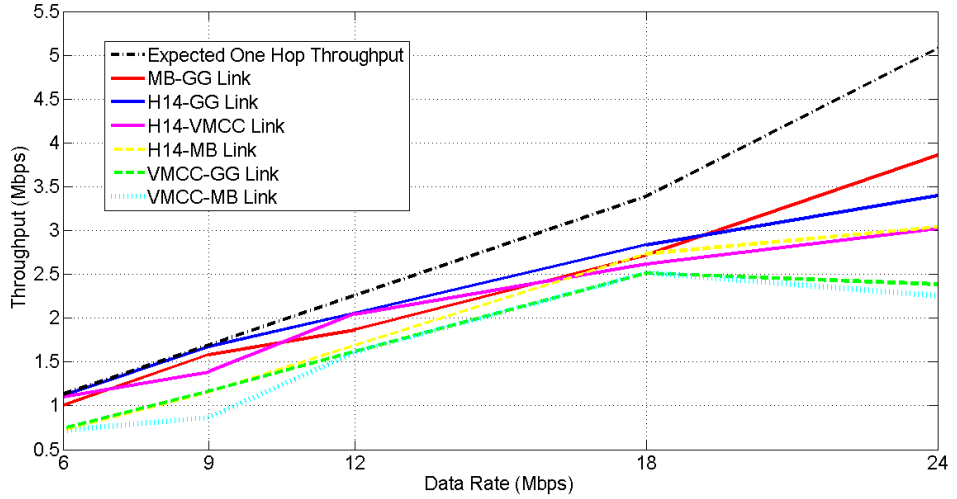

Figure 5e. TCP Throughput for various links in multi-hop topology as shown in Figure 4.

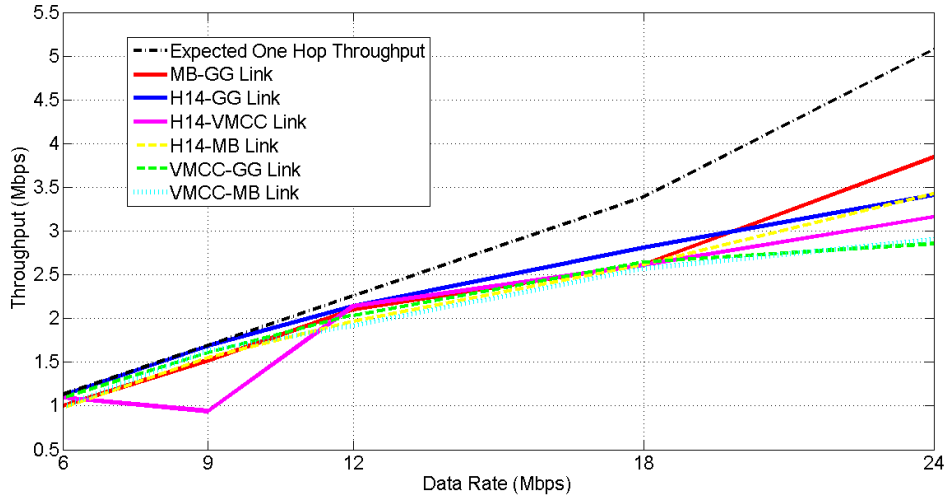

Figure 5f. UDP Throughput for various links in multi-hop topology as shown in Figure 4.

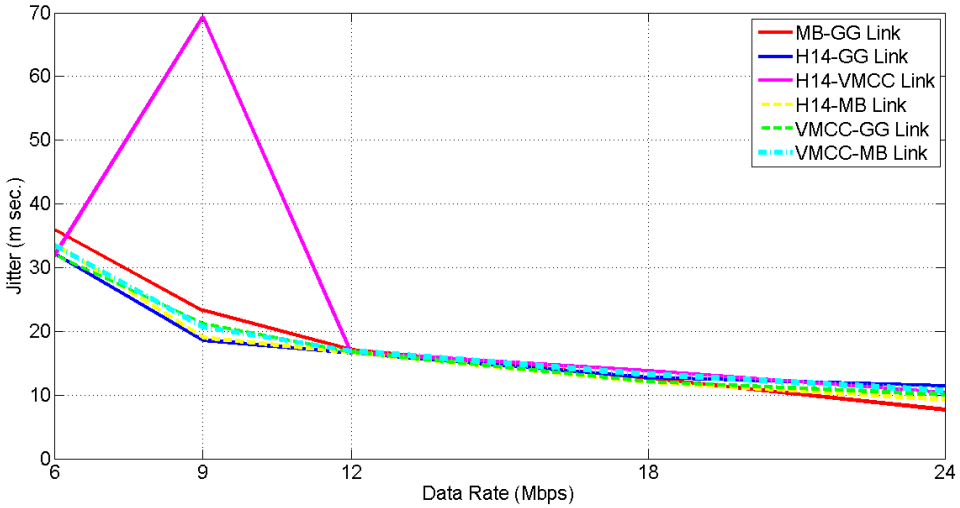

Figure 5g. UDP Jitter values for various links in multi-hop topology as shown in Figure 4.

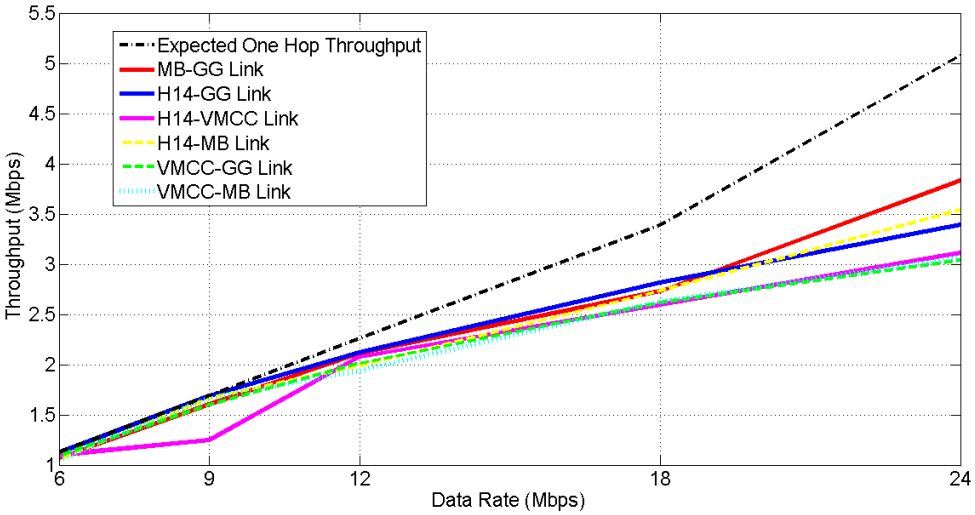

Figure 5h. UDP bidirectional throughput for various links in multi-hop topology as shown in Figure 4.

The TCP and UDP throughput measurements in the multi-hop topology show similar behaviour as compared with that in the point-to-multipoint topology. This could be attributed to the fact that the

allotment of time slots by the root node to all non-root nodes in the network is independent of the topology. Thus, the overall performance of the network comprising of LiTMAC based nodes would not depend on the exact topology by which the nodes connect to each other.

CSMA Performance

Next, we observe the throughput variations of CSMA MAC at different data rates. In order to make sure that the PHY layer data rates do not change in between the course of measurements, we set the data rate to fixed value before starting the measurements. The following set of graphs show the results obtained in TCP, UDP and UDP Bidirectional throughput measurements at different throughputs and in different links. Note that since we have performed the CSMA measurements only in the AP-Client mode (and not in ad-hoc mode with multi-hop topology), there are only one and two hop links in the network.

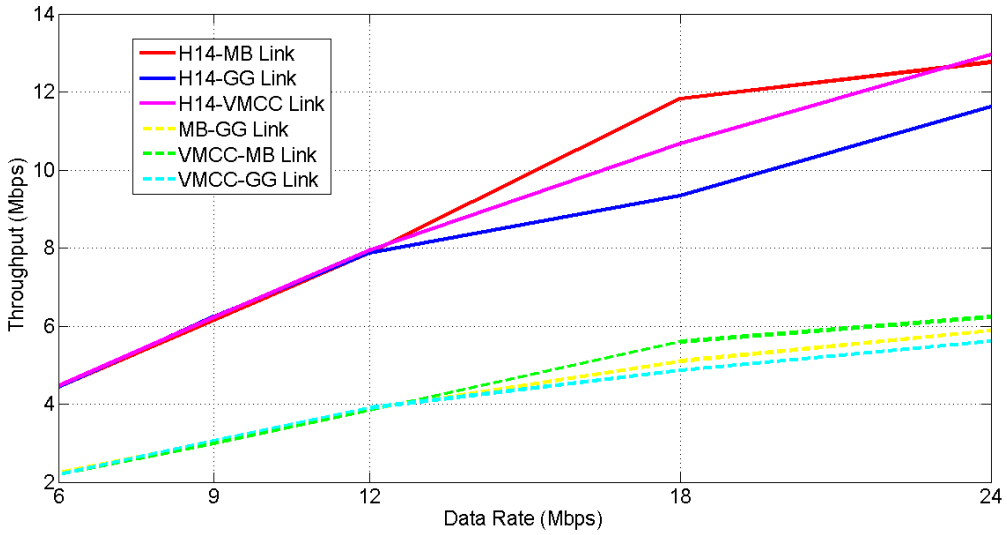

Figure 6a. TCP Throughput for various links in the AP-Client mode

It is clear from Figure 6a that the TCP throughput is maximum for single-hop links, i.e. AP-Client or Client-AP link. TCP throughput as high as 13 Mbps is observed for PHY layer data rate of 24 Mbps. On the other hand, if we consider a two hop link in this topology (Client1-AP-Client2), the throughput is almost halved. This is intuitive since both nodes try to gain access to the channel and the effective throughput at each client would be half of the case in which only client connected to the AP.

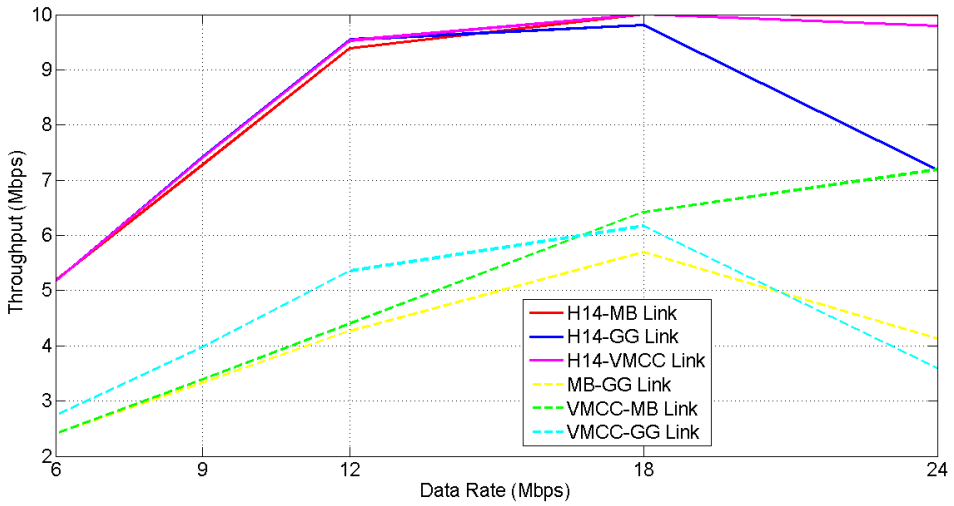

Figure 6b.UDP Throughput for various links in the AP-Client mode.

The maximum TCP throughput was observed at the PHY layer data rate of 24 Mbps. However, the maximum UDP throughput was observed at the PHY layer data rate of 18 Mbps. Moreover, the maximum UDP throughput (~ 10 Mbps) is lower than the maximum TCP throughput, which is not the case in TDMA based LiTMAC protocol.

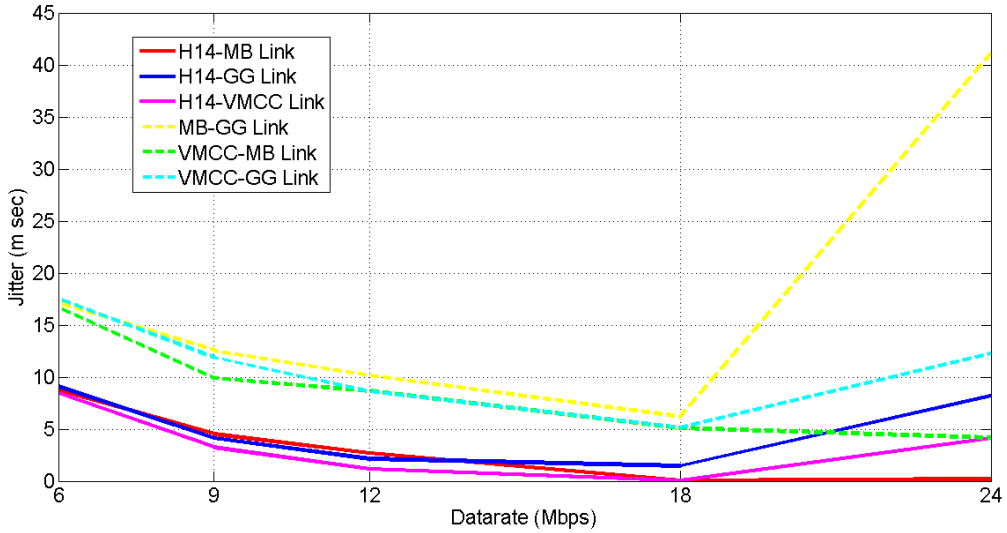

Figure 6c. UDP Jitter values for various links in the AP-Client.

At low data rates, the jitter values in case of CSMA MAC are significantly lower than that observed in the LiTMAC network. However, at higher rates, the jitter values are almost comparable to the LiTMAC network.

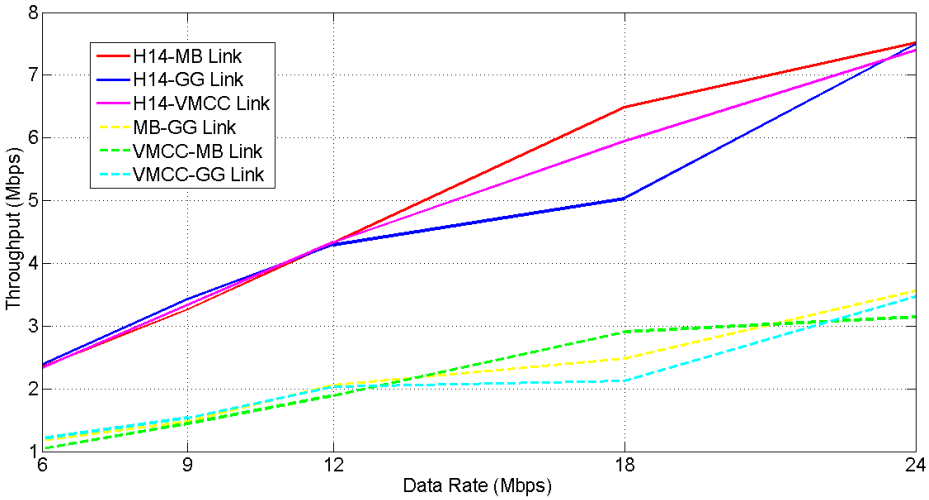

Figure 6c. UDP Bidirectional throughput for various links in the AP-Client.

If we compare the results of the UDP Bidirectional throughput measurements in the CSMA MAC v/s the TDMA based LiTMAC, we see that in LiTMAC all the links have almost same throughput, while in CSMA the one hop links have better throughput than that in LiTMAC. However, the two-hop links have poor throughput as compared to that of two-hop links in LiTMAC.

Effect of slot size on LiTMAC throughput

The expected UDP throughput calculated theoretically depends on the slot duration. In case of LiTMAC, a guard time of 100 microsec is provided in a slot. At low values of slot duration, this guard interval occupies a larger fraction of the slot, while at larger values of the slot duration the guard interval occupies a small fraction. Thus, it is expected as the slot size increases, the throughput would increase. These conclusions were drawn from [9].

To verify these observations, we fixed the PHY layer data rate at 24 Mbps in the multi-hop topology (Figure 4). We then performed pair-wise iperf throughput computation between all nodes by varying the slot size and keeping all other parameters constant. Figure 7a and 7b shows the effect of change in the slot size variation on the TCP and UDP throughput respectively. As described in [9], the UDP throughput indeed increases with increase in slot size albeit by a small value. However, the TCP throughput increases up to a particular value of the slot size and then decreases. This, again was highlighted in [9] and this decrease in throughput was attributed to the large waiting time for TCP Acks. The maximum observed TCP as well as UDP throughput was observed at a slot size of 10 msec.

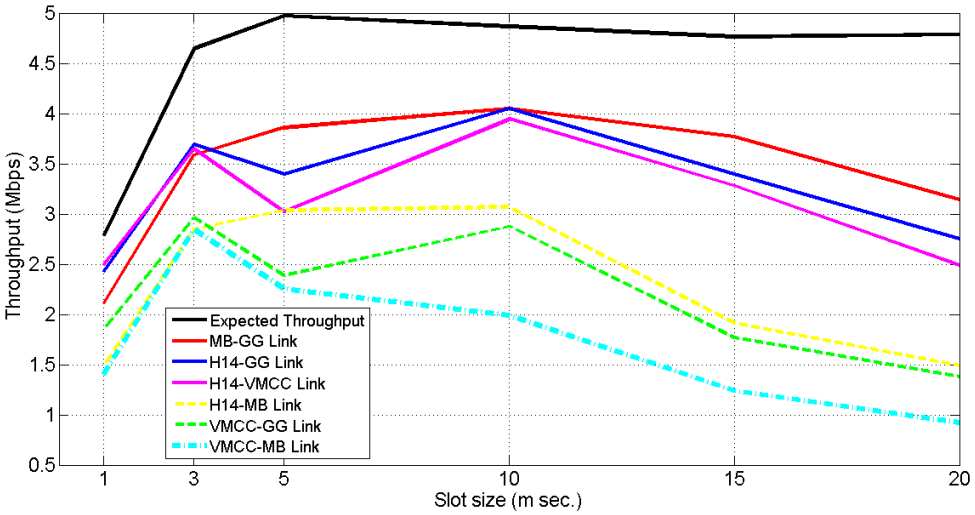

Figure 7a. TCP Throughput for various links in point-to-multipoint topology as shown in Figure 3.

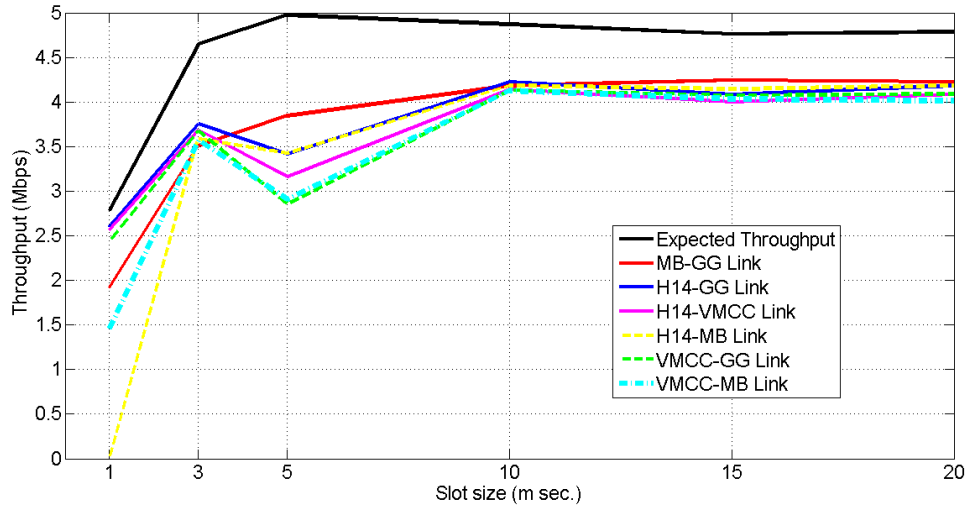

Figure 7b. TCP Throughput for various links in point-to-multipoint topology as shown in Figure 3.

Effect of Multiple traffic flows

One interesting observation is that in the given setting, CSMA performance far exceeds TDMA performance. One major reason for this is the fact that LiTMAC uses static TDM scheduling across the four nodes in the network. It assigns a slot to all nodes in every frame, whether or not the node actually has any data to transmit. In the measurements performed up to this point, at no point in time are all four nodes transmitting. This is because only one flow runs from an iperf client to an iperf server. The same thing happens in the CSMA nodes. Only one flow operates a time. As a result, there is not a lot of contention for the channel.

We now modify the measurement scenario and run the iperf server (TCP and UDP) on one node (H14) and run iperf clients at the remaining three clients (GG, MB and VMCC) in the point-to-multipoint topology in TDMA based network and in the CSMA network. We carried out the measurements in the TDMA based network for two different slot sizes, 5 msec and 10 msec. The mean aggregate throughput (and standard deviation) in case of CSMA devices, TDMA slot size 5 msec and TDMA slot size 10 msec are shown in the graph below. The choice of slot size 10 msec was

due to the fact that both the TCP and UDP throughput were maximum at the slot size of 10 msec as observed in Figures 7a and 7b. The other slot size was chosen as 5 msec as all our measurements were performed at the slot size of 5 msec.

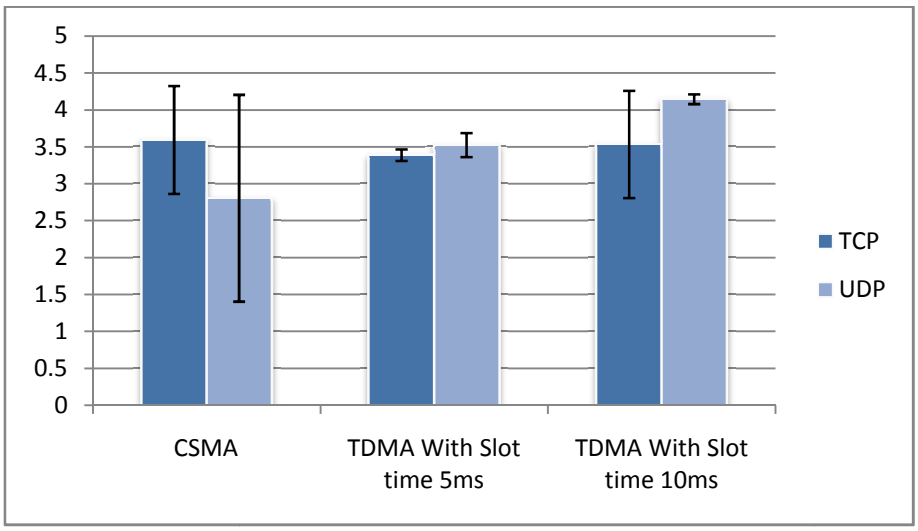

Figure 8: Effect of multiple flows running simultaneously on CSMA and LiTMAC performance

The y-axis represents the throughput in Mbits/sec. The vertical line on top of each bar denotes the standard deviation of the throughput. At the first glance itself, it is evident that in this scenario the CSMA performance and TDMA performance are comparable. The mean TCP throughput is almost the same for CSMA and TDMA with slot duration 5 msec and 10 msec. However, the standard deviation of the TCP throughput in CSMA is much larger than that in TDMA with 5 msec slot duration, while it is comparable to that in TDMA with 10 msec slot duration.

However, the major difference between the CSMA and TDMA MAC can be observed in case of UDP throughput in Figure 8. Not only is UDP throughput in CSMA less as compared to TDMA MAC, but also the standard deviation is much larger in case of CSMA than in TDMA based MAC. This suggests that when multiple flows exist in the network, the TDMA based MAC protocol can guarantee a certain level of throughput which is comparable to (if not more than) CSMA MAC. At the same time, the throughput is much more stable in the former case. This is critical in use cases like backhaul networking using wireless mesh networks.

Thus, while it may appear that CSMA based MAC protocol can provide higher system throughput than TDMA based MAC, when the number of nodes in the network increase CSMA performance degrades. We observed this in case of 1 AP and 3 clients. If more clients connect to the AP, then the system throughput would degrade further. On the other hand, in TDMA based MAC protocol, the system would provide a seemingly low, yet stable throughput to all users.

Chapter 5

Debugging

During the course of our work, we came across several issues related to compilation of the OS, configuring the boards, and field deployments. We list some of these problems in this chapter and the solutions we came across either via trial and error or via internet. Neither the list of problems, nor the solutions are exhaustive. This is just meant to look for some obvious solutions to frequently occurring problems.

Flashing Image

1. No Space left for loading kernel

While creating the ramdisk image, the default space provided for /dev/mtdblock1 is 4MB. If the kernel size (size of file openwrt-ar71xx-vmlinux.elf) exceeds 4MB, then this error will pop up.

Solution: Provide additional space for the /dev/mtdblock1 partition. We provided 8MB and the issue was resolved.

Software Issues

1. Command 'iwconfig' not found

This problem could occur particularly in case openwrt attitude adjustment. While the

Laboratory Testing

1. Wireless Interface not running

a) Make sure the wireless interface has been enabled

Comment/delete the line 'option disable 1' in the /etc/config/wireless file

b) Make sure you are using the correct wireless interface.

A mixture of RB411AR and RB433AH boards has been used for deploying the network. While the RB433AH board does not have any internal wireless card, the RB411AR board has an internal 2.4GHz wifi card. However, we have not made use of this interface in our deployment. In the /etc/config/wireless file, this internal wireless interface will be listed as the first device (wlan0 or wifi0). Make sure that you have enabled the correct external wireless interface.

2. Two boards are created as per the instructions provided in Section xxx. However, the two cards do not ping each other.

a) One possible reason for this problem is incorrect timing granularity. As previously stated, the granularity in LiTMAC is of 1 msec, as the Timer Resolution was set to 1000 Hz. If, however, the timing resolution was not set using 'make kernel_menuconfig', then this problem will arise.

Solution: Recompile the openwrt kernel image and the rootfs file after setting the Timing Resolution to 1000 Hz, and flash it on to the RouterBoard.

b) For two nodes to connect (ping) to each other, the radio interface must

- Have the same SSID
- Operate on the same channel (same frequency)
- Operate in the same mode

These parameters can be viewed using the 'iwconfig' command. Make sure both nodes have the same values of these parameters and try again.

c) Check if the Network interface for the wireless card in '/etc/config/network' is configured to operate in bridge mode. If so, comment or delete the line and reboot the system.

d) The TDMA specific configurations in LiTMAC must be done manually. As described in section xxx, this is done using files 'config.cfg' and 'startup.sh'. The file startup.sh is executed during bootstrapping.

Solutions:

- Make sure the startup.sh file gets executed when the RouterBoard reboots. For example, insert a line at the beginning of the file to echo a string or create a file and check if this is executed. If not, check if the path of the startup.sh file is mentioned in the '/etc/rc.d/S99sysctl' file.
- Check if the contents of the 'config.cfg' file is pushed in to the '/proc/net/madwifi/ath0/fractel_config' file.
    - If the contents are being pushed but there are any spaces (' ') in between the parameters and the values, then remove these spaces and try again. For instance, if the fractal data rate is set as 'fractal_data_rate = 10', rewrite it as 'fractal_data_rate=10' and try again.
    - If the contents are not being pushed, one possible reason is lack of available memory on the device.
    - Check the available size in '/dev/root' partition by using the command 'df'
    - If the memory available is indeed less, delete the large sized files (particularly the log files of fractal_readlog and readlog), and reboot the system.
    - Once the system reboots check the available memory once again using 'df'. If, however, the memory available is the same as before, then there is some problem with the RouterBoard. We were unable to figure out this problem, and replaced the board.

3. No memory available in the device

This problem might occur frequently while using LiTMAC. The default startup.sh runs the fractal_readlog command every time the device reboots and stores the result in /root/outfile. The size of the file is extremely large. Try deleting this file and check the memory available using the 'df' command. If the problem persists, then as stated above, we replaced the boards.

Chapter 6

Conclusions and Future Work

In our work we tried to compare the performance of a CSMA and TDMA based MAC protocol. Our experiments were performed in a limited set of environments and several factor which could effect the network performance in practical deployment scenarios might not have been accounted for in our settings. However, based on the results we observed during our measurements, we can conclude the following.

The LiTMAC is inefficient in two respects as far as we observed.

- Firstly, it does not perform Link layer fragmentation. As a result, in every time slot some portion of the air-time is unutilized and this leads to inefficient utilization of the slot. To counter this, packet size can be optimized in case of iperf UDP measurements and an ideal packet size can be obtained that maximized the UDP throughput. However, in practical deployment scenarios, this is highly impractical as user data is bursty and in different sizes.
- Secondly, the slot allocation to the different nodes in the network is static. Each node in the network is allocated a time slot, whether or not it has data to send. This results in wastage of the slot if no data has to be sent by the respective node. As a result, in scenarios where not all the nodes in the network have data to send continuously, CSMA will outperform LiTMAC.

Advantages gained due to the use of LiTMAC are

- LiTMAC provides stable throughput to all nodes in the network. This is in contrast to the CSMA based MAC protocols which are inherently unfair to the 'good' users. A few nodes with poor channel conditions can bring down the aggregate throughput of the network significantly in case of CSMA based network.
- The performance of LiTMAC is stable as compared to CSMA. As seen in the previous chapter, in conditions suitable for LiTMAC, it provides throughput almost comparable to CSMA based protocols while guaranteeing more stability in providing the said throughput.

One interesting observation in our experiments was that as compared to 2.4 GHz WiFi, the performance of CSMA protocol was better than LiTMAC in larger set of circumstances in the sub-GHz band. Though we have not carried out experiments in the 2.4 GHz band, these are observations based on comparing our results with those obtained in [10]. We attribute this improved performance to the non-interference from other CSMA nodes in the sub-GHz band.

As part of the future work, we intend to perform link characterization in different sets of environments, one of which being the current deployment topology. However, to do this, we first need to characterize the hardware at the transmitter as well as the receiver side. The objective of this work is to compare the results to existing empirical models and determine their suitability in the Indian terrain conditions.

References

[1] S. Mishra, A. Sahai, 'How much white space is there?,' Tech. Report UCB/EECS-2009-3, EECS Department, UC Berkeley, Jan. 2009.

[2] M. Nekovee, 'Quantifying the Availability of TV White Spaces for Cognitive Radio Operation in the UK,' in Proc. Of IEEE Intl. Conf. On Communications Workshops, June 2009.

[3] J. Van de Beek et. al., 'TV White Space in Europe,' in IEEE Trans. On Mobile Computing, vol. 11, no. 2, Feb. 2012, pp 178-188.

[4] T. Shimomura, T. Oyama and H. Seki, 'Analysis of TV White Space Availability in Japan,' in Proc. of IEEE Vehicular Tech. Conf., Sep. 2012

[5] G. Naik, S. Singhal, A. Kumar, A. Karandikar, 'Quantitative Assessment of TV White Space in India,' in Proc. Of National Conference on Communications, March 2014.

[6] K. Paul, A. Varghese, S. Iyer, B. Ramamurthi, A. Kumar, 'WiFiRe: Rural Area Broadband Access Using the WiFi PHY and a Multisector TDD MAC,' IEEE Communications Magazine, pp. 111-119, Jan. 2007.

[7] S. Sen, B. Raman, 'Long Distance Wireless Mesh Network Planning: Problem Formulation and Solution,' The 16th Annual Interntional World Wide Web Conference (WWW 2007), May 2007.

[8] Web Link: http://www.doodlelabs.com/products/radio-transceivers/sub-ghz-range/470-790-mhz-tvws-100/

[9] A. Dhekne, N. Uchat, B. Raman, 'Implementation and Evaluation of a TDMA MAC for WiFi-based Rural Mesh Networks'.

[10] V. Sevani, 'Improving Performance In TDMA Over WiFi-PHY Mesh Networks for Rural Internet Connectivity,' Ph. D. Thesis.