

A Tutorial on Broadcasting Packets over Multiple-Channels in a Multi-Interface Network Setting in NS-2

Zeeshan Ali Khan and Mubashir Husain Rehmani *

Version 1

5th July 2010

*Z. A. Khan is a PhD student at LEAT, CNRS, University of Nice Sophia Antipolis, Sophia Antipolis, France e-mail: zkhan@unice.fr; M. H. Rehmani is a PhD student at Lip6/Université Pierre et Marie Curie (UPMC) – Paris-6, France, e-mail: mubashir.rehmani@lip6.fr. The author M. H. Rehmani would like to thank A. C. Viana, who is with INRIA, France email: aline.viana@inria.fr

Contents

1	Introduction	3
2	Methodology	3
2.1	Modifications in TCL file test4wcett.tcl	4
2.2	How to Enable Hello Packets	4
2.3	How to Send Hello Packets on a Specific Channel over Multiple Interfaces	4
2.4	Ways of Selecting the Channel	5
2.5	Modifications in wireless-phy.cc file	6
3	Conclusion	6

Abstract

With the proliferation of cheaper electronic devices, wireless communication over multiple-channels in a multi-interface network is now possible. For instance, wireless sensor nodes can now operate over multiple-channels. Moreover, cognitive radio sensor networks are also evolving, which also operates over multiple-channels. In the market, we can find antennas that can support the operation of multiple channels, for e.g. the cc2420 antenna that is used for communication between wireless sensor nodes consists of 16 programmable channels. The proper utilization of multiple-channels reduces the interference between the nodes and increase the network throughput. Recently, a Cognitive Radio Cognitive Network (CRCN) patch for NS-2 simulator has proposed to support multi-channel multi-interface capability in NS-2.

In this tutorial, we consider how to simulate a multi-channel multi-interface wireless network using the NS-2 simulator. This tutorial is targeted to the novice users who want to understand the implementation of multi-channel multi-interface in NS-2. We take the Cognitive Radio Cognitive Network (CRCN) patch for NS-2 simulator and demonstrate broadcasting over multiple-channels in a multi-interface network setting. In our setting, node broadcasts the Hello packets to its neighbors. Neighboring nodes receive the Hello packets if and only if they are tuned to the same channel. We demonstrate through example that the tuning of receivers can be done in two fashions.

1 Introduction

The Network Simulator (NS-2) [1] is a most widely used network simulator. This tutorial uses the implementation of WCETT [2] based multi-channel wireless mesh protocol in the Cognitive Radio Cognitive Network (CRCN) [3] patch developed for NS-2. The expected audience are students who want to understand the broadcasting mechanism in multi-channel multi-interface environment in NS-2. The version considered is NS-2.32 and 2.33, but it might be useful to other versions as well. Throughout the rest of this tutorial, the under considered files are `wcett.cc`, `wcett.h`, `wcett_logs.cc`, `wcett_packet.h`, `wcett_rqueue.cc`, `wcett_rqueue.h`, `wcett_rtable.cc`, `wcett_rtable.h`, which can be found in WCETT folder in the NS-2 base directory. The tcl file `test4wcett.tcl` can be found in the NS-2 based directory. Other relevant files are `phy.cc` and `wireless-phy.cc`.

2 Methodology

Our goal is to broadcast Hello packets over multiple interfaces. The methodology we adopted is to use Hello packets and broadcast them over multiple interfaces. We then demonstrated that a specific channel can be assigned to a particular interface. This assigned can be done either directly at the lower layer or can be communicated from the routing layer to the lower layer. We now describe the relevant section of codes.

2.1 Modifications in TCL file test4wcett.tcl

We use the TCL file test4wcett.tcl present in the NS-2 folder. WCETT is used as a routing protocol, whose code is quite similar to AODV and having the additional capability of calculating WCETT metric and can be able to pass channel selection information to lower layers. IEEE 802.11 mac is selected as mac protocol. The total number of interfaces of each node is selected as 3 in the tcl script:

```
set val(ni) 3;
```

2.2 How to Enable Hello Packets

By default HELLO packets are disabled in the wcett protocol. To enable broadcasting of Hello packets, comment the following two lines present in wcett.cc

```
#ifndef WCETT_LINK_LAYER_DETECTION
```

```
#endif LINK_LAYER_DETECTION and recompile NS-2 by using the following commands on the terminal:
```

```
make clean
```

```
make
```

```
sudo make install
```

2.3 How to Send Hello Packets on a Specific Channel over Multiple Interfaces

```
2025
2026 if(nifaces)
2027 {
2028     //carmen
2029     for(int i=0;i<nIfaces;i++)
2030     {
2031         Packet *p_copy=p->copy();
2032         struct hdr_cmn *ch_copy = HDR_CMN(p_copy);
2033         ch_copy->channelindex_ = 1;
2034         printf("SendHello from index: %d using channel: %d interface: %d \n", index, ch_copy->channelindex_, i);
2035         Scheduler::instance().schedule(targetlist[i], p_copy, 0.0);
2036     }
2037 }
2038 else
2039 {
2040     Scheduler::instance().schedule(target_, p, 0.0);
2041 }
2042 }
2043
2044
```

Figure 1: SendHello() function in the wcett.cc file.

In order to send Hello packets on a specific channel over multiple interfaces, we need to assign the specific channel. This assignment of channel can be done by using the following variable:

```
ch_copy->channelindex_
```

This channel value will then be used by the network interface (phy.cc file) to select a channel for sending the packets. We can check how packets are transmitting on a specific channel over multiple interfaces by adding some lines in SendHello() function of wcett.cc file (cf. Fig. 1). In Fig. 1, the lines from 2026 to 2041 send Hello packets over multiple interfaces. Note that if we do not specify the channel selection explicitly, as we mentioned on line 2033 (channel # 1), nodes will send the packet automatically on the channel decided at the lower layers.

We can also see how nodes receive the hello packets by using printf command in recvHello() function present in the wcett.cc file (cf. Fig. 2).

```

2067
2068 //record the hellocounter for each neighbor on each channel
2069 if(nIfaces){
2070     int iface=HDR_CMN(p)->localif();
2071     ++ChannelETT[rp->rp_dst][iface];
2072     printf("recvHello at: %d using channel: %d and interface: %d\n", index, chtemp->channelindex_, iface);
2073
2074 }
2075

```

Figure 2: RecvHello() function in the wcett.cc file.

2.4 Ways of Selecting the Channel

There are two ways to select the channel value:

1. The routing layer decides the channel and communicates this information to the lower layer. Note that this is done by adding a field in the packet header. The lower layer can use this channel decision value from the packet header. This channel selection decision value can be used to select the channel of the current node by making the following change in phy.cc file at the beginning of its recv() function:

```
nchannel = hdr->channelindex_;
```

which changes the current channel value to equal *channelindex_* value passed from WCETT routing layer or simply wcett.cc.

2. We can assign some channels to some specific nodes. For e.g. we can select the channel 0 for node 7's interfaces and for all other nodes, the selected channel value is 1. This could be done by adding the following lines at the start of the recv() function in phy.cc

```
if (node()-> nodeid() == 7)
```

```
nchannel = 0;
```

```
else
```

```
nchannel = 1;
```

If we make the above changes, channel 0 is assigned to node 7, while channel 1 is assigned to all the remaining nodes.

2.5 Modifications in wireless-phy.cc file

In order to tune an interface based on the channel value passed from the routing layer (i.e. wcett.cc), we are required to make the following change in sendUp function of wireless-phy.cc file. The lines 430-431 (cf. Fig. 3) tunes the interface to the channel decision communicated from the routing layer.

```

429
430         if (nchannel != hdr->channelindex_)
431             goto DONE;
432
433
434
435         /*
436          * The MAC layer must be notified of the packet reception
437          * now - ie; when the first bit has been detected - so that
438          * it can properly do Collision Avoidance / Detection.
439          */
440         pkt_rcvd = 1;
441         //carmen receive traffic, increase traffic count
442         ((MobileNode*)node())->PacketPr_>increaseTrafficCount();
443
444     DONE:
445         p->txinfo_.getAntenna()->release();
446
447         /* WILD HACK: The following two variables are a wild hack.
448          * They will go away in the next release...
449          * They're used by the mac-802.11 object to determine
450          * capture. This will be moved into the net-if family of
451          * objects in the future. */
452         p->txinfo_.RxPr = Pr;
453         p->txinfo_.CPThresh = CPThresh_;
454         //carmen
455         int currentnode;
456         currentnode=node()->nodeid();
457

```

Figure 3: SendUp() function in the wireless-phy.cc file.

3 Conclusion

In this tutorial, we showed that how broadcasting of packets can be done in multi-interface network setting in NS-2. Moreover, we also demonstrated through a working example that an interface can be tuned to a particular channel based on the channel value passed from the routing layer (that is wcett.cc). One can also perform complete broadcast by broadcasting over all the available channels.

References

- [1] [Online]. Available: <http://www.isi.edu/nsnam/ns/>
- [2] J. P. Richard Draves and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," *MobiCom'04*, 2004.
- [3] [Online]. Available: <http://stuweb.ee.mtu.edu/~ljalian/>