Remote Management Guide for Smart Radio

Advanced Mesh Router for Private Wireless Networks

Introduction

The Smart Radio runs the Mesh Rider OS. It is a customized version of Openwrt with enhancements useful for applications requiring low-latency command-and-control transmission and high-throughput video - e.g. UAV and robotics.

The purpose of this guide is to aide a user in remotely configuring Smart Radio settings. There are four ways to configure the Smart Radio.

1. Using the Web GUI
2. Using SSH
3. Using the JSON-RPC API
4. Using SNMP

A full technical discussion of all of the different configuration parameters of the Openwrt system is beyond the scope of this guide, and the user is encouraged to explore the OpenWrt website.

Web GUI

The Web GUI can be accessed in any web browser at https://<IP ADDRESS> (port 443). Note that the web browser uses a self-signed certificate. This means that connection to the web browser is encrypted, but not authenticated. The first time you access the Smart Radio from a new browser, you will get a SSL certificate warning. It is okay to ignore the warning and proceed.

SSH

It is possible to remotely retrieve network information using the GUI by logging into the Smart Radio with a web browser at the ETH0 IP address, or over the command line using a combination of SSH and Linux command-line utilities. Please refer to Appendix A for some examples.

JSON-RPC API

If you plan on integrating remote network retrieval into an application, then the JSON-RPC API may be useful. For a typical application, there are two types of interactions you would normally make with the Smart Radio. Firstly you may want to reconfigure certain basic operating parameters such as the operating channel or operating bandwidth. Secondly, you may want to retrieve current network status such as RSSI. In the first case, we recommend you use the UCI system, and for the second case, you will need to execute specific programs in the radio such as iwinfo, iw, or batctl.

For detailed descriptions on how to use the JSON-RPC API, please refer to the Appendix B.

SNMP

The Smart Radio also supports the SNMP protocol. Please refer to Appendix C for more details on using the SNMP.

Appendix A – Using SSH

The followings are some examples of remotely executing a command via SSH to obtain network information from the node.

ssh root@10.223.134.70 ‘iw wlan0 info’

Lists the current configuration of wlan0

ssh root@10.223.134.70 ‘iw wlan0 station dump’

shows connection information to all currently connected stations.

ssh root@10.223.134.70 ‘iw wlan0 list’

shows the capabilities of the wlan0 interface. An example output is shown below.

```
root@LEDE:~# iw dev wlan0 info
Interface wlan0
        ifindex 13
        wdev 0x7
        addr 00:30:1a:4e:86:46
        type mesh point
        wiphy 0
        channel 12 (915 MHz), width: 20 MHz, center1: 915 MHz
        txpower 32.00 dBm
```

Appendix B – Using the JSON-RPC API

For detailed descriptions on how to use the JSON-RPC API, please refer to the following links:

https://openwrt.org/docs/techref/ubus

https://github.com/openwrt/luci/wiki/JsonRpcHowTo

The listings below, along with additional scripts and packages can be found here:

https://doodlelabs.sharepoint.com/:f:/s/Technical/Elhb0AcP3ClCltsuPOu9gqEBY0v34XZYp1GDrMCTaRS8oA?e=IgmAJd

In order to access the radio, you first need to set the permissions for your user profile (to be created next). Create a new profile /usr/share/rpcd/acl.d/superuser.json like the one below.

```
{
	"superuser": {
		"description": "Super user access role",
		"read": {
			"ubus": {
				"*": [ "*" ]
			},
			"uci": [ "*" ],
			"file": {
				"*": ["*"]
			}
		},
		"write": {
			"ubus": {
				"*": [ "*" ]
			},
			"uci": [ "*" ],
			"file": {
				"*": ["*"]
			},
			"cgi-io": ["*"]
		}
	}
}
```

Next, create a user profile in /etc/config/rpcd, and then restart the process with /etc/init.d/rpcd restart. An example profile is shown below.

```
config login
        option username 'root'
        option password '$p$root'
        list read 'superuser'
        list write 'superuser'
```

The API is now ready to be used. You should adjust the user profile and permissions to your liking. The permissions above could pose a security risk. In order to use API, you need to get a session ID and use it for subsequent requests. For ubus, for example, you can try the following call,

```
curl -k https://<IP-ADDRESS>/ubus -d '
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "call",
  "params": [ "00000000000000000000000000000000", "session", "login", { "username": "root", "password": "secret"  } ]
}'
```

the -k option is required because the Smart Radio doesn't use a third party certificate authority. An example of using JSON-RPC API for file access is shown below. Substitute <TOKEN> with the value returned above.

```
curl -k https://<IP-ADDRESS>/ubus -d '
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "call",
  "params": [ “<TOKEN>”, "file", "read", { "path": "/etc/board.json" } ]
}'
```

Appendix C – Using SNMP

Before you can use SNMP, you need to open up the firewall at port 161 for UDP traffic. In the GUI, navigate to `Network Configuration → Firewall`. Under the section `Open ports on router`, add a new rule like that shown in the figure below. Click `Add`, and after the page reloads, click `Save & Apply`.

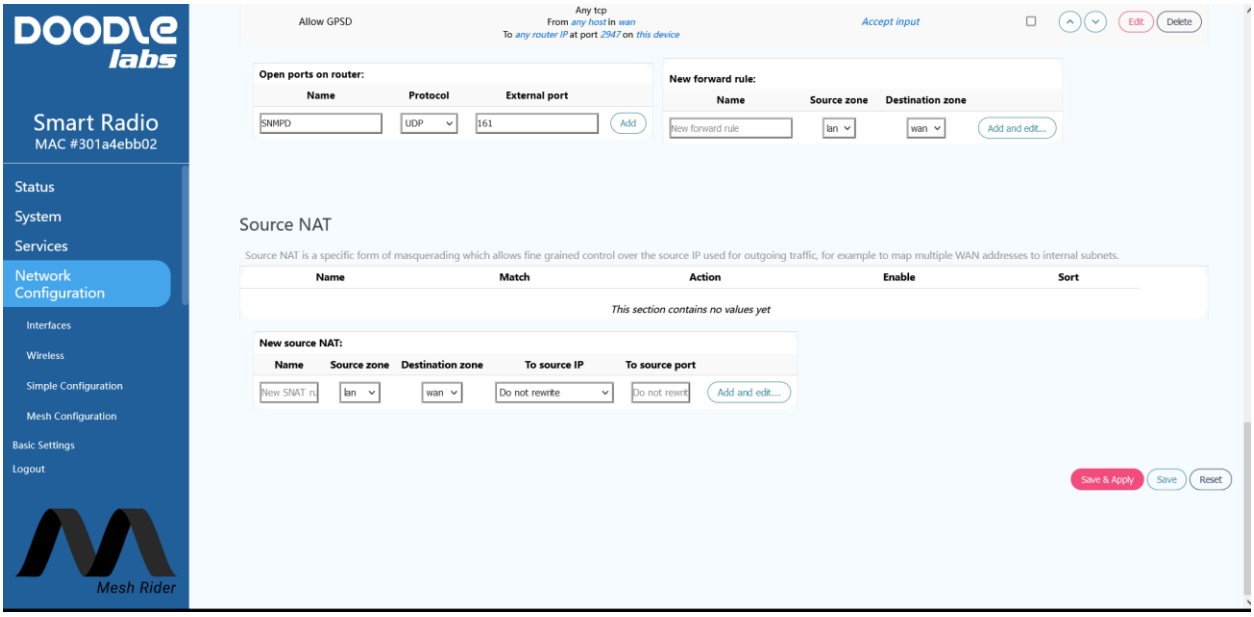

Fig. 1 – Opening the Firewall for SNMP

In order to enable SNMP, click the `Advanced Settings` button in the bottom left corner of the GUI. Then navigate to `services → SNMPD`. Uncheck `Disable SNMPD Service`, and fill in the table. Finally click `Save & Apply`.

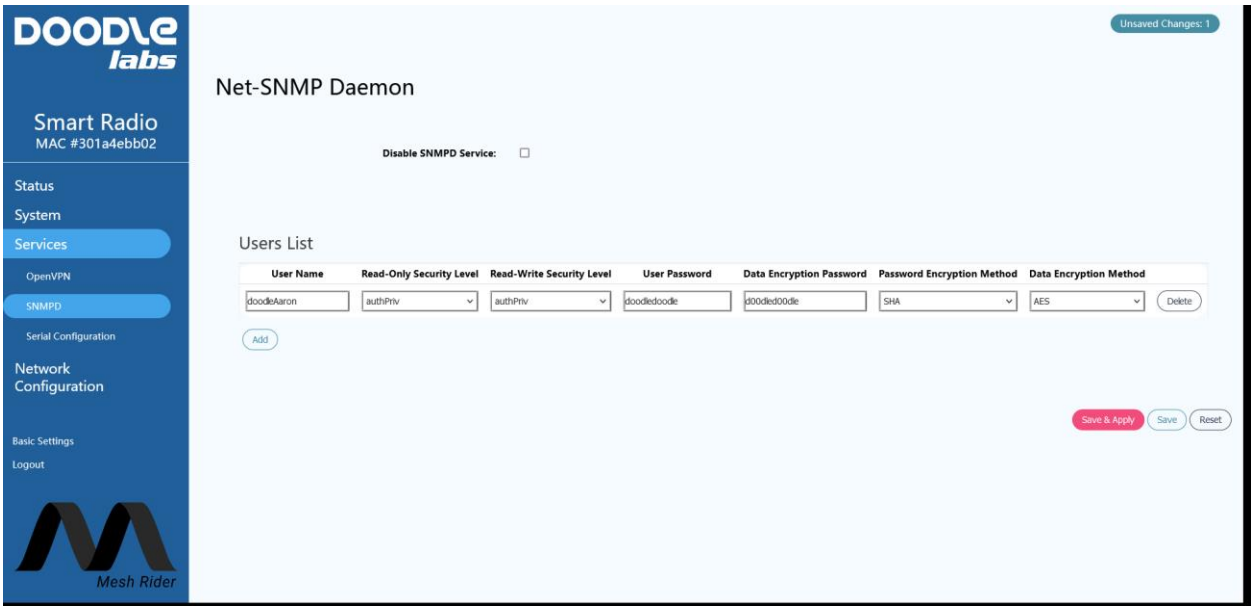

Fig. 2 – Enabling and Configuring SNMP

With SNMP enabled, we can test it using the program `snmpwalk` in Linux. The listing below shows an example test script.

```
#!/bin/sh
IPADDR="10.223.187.2"
USRNAME="doodleAaron"
USRPASS="doodledoodle"
DATAPASS="d00dled00dle"

snmpwalk -m ALL -M ./ -v 3 -u $USRNAME -l authPriv -a SHA -A $USRPASS -x AES
-X $DATAPASS $IPADDR 1.3.6.1.3.12314
```

Modify the user login information in `snmpwalk.sh` using the information you used in the web GUI setup. The MIB is defined in /usr/lib/lua/smithsnmpd/mibs/wifi.lua and needs to be modified to be able to support the information defined in WIFI-MIB.txt which is available here:

https://doodlelabs.sharepoint.com/:f:/s/Technical/Elhb0AcP3ClCltsuPOu9gqEBY0v34XZYp1GDrMCTaRS8oA?e=IgmAJd

Appendix D – Common CLI Commands

This section provides commands commonly used in the Smart Radio for configuration and diagnostics.

UCI

The UCI system is used for configuration. Most UCI files are found at `/etc/config/`. This is a slow method of configuration, but changes are saved over a reboot. After committing changes, it is necessary to restart the relevant service (see the section below).

Command	Example Output	Purpose
uci show	TOO LONG	Shows the full UCI configuration including all sections
uci show wireless	wireless.radio0=wifi-device wireless.radio0.type='mac80211' wireless.radio0.hwmode='11g' wireless.radio0.path='platform/qca953x_wmac' wireless.radio0.htmode='HT20' wireless.radio0.fes_disabled='0' wireless.radio0.rxantenna='1 2' wireless.radio0.txantenna='1 2' wireless.radio0.channel='7' wireless.radio0.chanbw='15' wireless.radio0.disabled='0' wireless.radio0.legacy_rates='0' wireless.radio0.distance='4000' wireless.wifi0=wifi-iface wireless.wifi0.device='radio0' wireless.wifi0.mode='mesh' wireless.wifi0.network='mesh_dev' wireless.wifi0.mesh_id='simpleconfig' wireless.wifi0.mesh_fwding='0' wireless.wifi0.mesh_nolearn='1' wireless.wifi0.mesh_ttl='1' wireless.wifi0.mcast_rate='12000' wireless.wifi0.encryption='psk2+ccmp' wireless.wifi0.key='DoodleSmartRadio'	Shows the wireless configuration. You can further filter the output with “uci show wireless.radio0” for example.
uci set wireless.radio.chanbw=5	NONE	Sets the channel bandwidth to 5 MHz
uci set wireless.radio0.txantenna=’1’	NONE	Sets the transmit antenna to Antenna 0 ONLY
uci commit	NONE	Saves changes. You can save individual sections too. For example “uci commit wireless”. After committing changes, you need to restart the service.

Restarting a service

After making configuration changes, restart the relevant service.

Command	Example Output	Purpose
ls /etc/init.d	RUN COMMAND	See a list of services to enable, restart, stop etc.
/etc/init.d/network restart **Available commands: restart/start/stop/enable/disable/reload**	NONE	Restarts the networking. Required after making changes to the network, like IP addressing. This will also restart other networking services like the firewall, dhcp, and the wireless interfaces.
wifi	NONE	Restarts the wireless interfaces
/etc/init.d/firewall restart	NONE	Restarts the firewall. Required after making changes to the firewall, like opening a network port.
reboot	NONE	Reboots the radio

Real-Time Configuration

The commands below work on-the-fly, but do not survive a network restart or reboot.

Command	Example Output	Purpose
iw wlan0 set tx power fixed 2000	NONE	Sets the TX power to 20 dBm. Note that the power is measured in millibels, so divide by 100 to get decibels.
iw wlan0 set tx power auto	NONE	Sets the TX power to auto (highest power)
iw wlan0 set bitrates ht-mcs-2.4 <RATE>	NONE	Fixes the bitrate. <RATE> is the MCS rate between 0 and 15 where 0-7 are single-stream rates, and 8-15 are dual-stream rates.
iw wlan0 set bitrates	NONE	Sets the MCS rate to auto
iw dev wlan0 mesh chswitch <CHANNEL> <# BEACONS>	NONE	Sends a channel switch announcement to all MESH nodes to switch to <CHANNEL> after sending <# BEACONS> beacons. CURRENTLY BUGGED (August 2021) and only the local radio will switch frequency.
hostapd_cli chan_switch <# BEACONS> <FREQ> ht	Selected interface 'wlan0' OK	Sends a channel switch announcement to all WDS Client nodes to switch to <FREQ> after sending <# BEACONS> beacons.

Getting Connection Information

Command	Example Output	Purpose
iw wlan0 station dump **OR** **iw wlan0 station get <MAC>**	Station 00:30:1a:4e:bb:26 (on wlan0) inactive time: 70 ms rx bytes: 2009144 rx packets: 18003 tx bytes: 91052 tx packets: 404 tx retries: 81 tx failed: 0 rx drop misc: 47 signal: -45 [-46, -49] dBm signal avg: -45 [-46, -50] dBm Toffset: 69680156242 us tx bitrate: 58.5 MBit/s MCS 6 rx bitrate: 52.0 MBit/s MCS 11 …	Gets information about all connected stations or an individual station. Expected throughput is not accurate.
iw wlan0 station get <MAC> | grep "inactive time"	inactive time: 70 ms	Shows how long it has been since a station was last seen
iw wlan0 station get <MAC> | grep "signal: "	signal: -45 [-46, -49] dBm	Shows the RSSI for a particular station
iwinfo wlan0 assoclist	00:30:1A:4E:F3:00 -36 dBm / -95 dBm (SNR 59) 90 ms ago RX: 29.2 MBit/s, MCS 4, 15MHz 34646 Pkts. TX: 39.0 MBit/s, MCS 11, 15MHz 510 Pkts. expected throughput: 27.3 MBit/s 00:30:1A:4E:BB:25 -51 dBm / -95 dBm (SNR 44) 100 ms ago	Information on all associated stations

	RX: 87.7 MBit/s, MCS 14, 15MHz 32927 Pkts. TX: 39.0 MBit/s, MCS 5, 15MHz 796 Pkts. expected throughput: 27.3 MBit/s	
iw wlan0 info	Interface wlan0 ifindex 24 wdev 0x3 addr 00:30:1a:4e:bb:09 type mesh point wiphy 0 channel 7 (2442 MHz), width: 15 MHz, center1: 2442 MHz txpower 36.00 dBm …	Get information about the current wireless settings
iw wlan0 survey dump	… Survey data from wlan0 frequency: 2442 MHz [in use] noise: -95 dBm channel active time: 52516424 ms channel busy time: 11001429 ms channel receive time: 8834551 ms channel transmit time: 388874 ms …	Get channel usage statistics
batctl	RUN APPLICATION	See a list of commands for mesh interface information and configuration
batctl o	[B.A.T.M.A.N. adv 2021.0-openwrt-1, MainIF/MAC: wlan0/00:30:1a:4e:bb:09 (bat0/8a:0a:22:0a:2e:58 BATMAN_IV)] Originator last-seen (#/255) Nexthop [outgoingIF] 00:30:1a:4e:f3:00 0.000s (187) 00:30:1a:4e:bb:27 [wlan0] 00:30:1a:4e:f3:00 0.000s (178) 00:30:1a:4e:bb:0a [wlan0] * 00:30:1a:4e:f3:00 0.000s (243) 00:30:1a:4e:f3:00 [wlan0] 00:30:1a:4e:bb:0a 0.000s (184) 00:30:1a:4e:bb:27 [wlan0] 00:30:1a:4e:bb:0a 0.000s (184) 00:30:1a:4e:f3:00 [wlan0] * 00:30:1a:4e:bb:0a 0.000s (243) 00:30:1a:4e:bb:0a [wlan0] * 00:30:1a:4e:bb:27 0.080s (253) 00:30:1a:4e:bb:27 [wlan0] 00:30:1a:4e:bb:27 0.080s (184) 00:30:1a:4e:bb:0a [wlan0] 00:30:1a:4e:bb:27 0.080s (180) 00:30:1a:4e:f3:00 [wlan0]	See information about connected mesh nodes including preferred hop (*), last-seen time, transmit quality (#/255)…

Networking Information

Command	Example Output	Purpose
ifconfig br-wan	br-wan Link encap:Ethernet HWaddr 00:30:1A:4E:AA:09 inet addr:10.223.187.9 Bcast:10.223.255.255 Mask:255.255.0.0 inet6 addr: fe80::230:1aff:fe4e:aa09/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:113310 errors:0 dropped:0 overruns:0 frame:0 TX packets:105496 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:36813834 (35.1 MiB) TX bytes:28067314 (26.7 MiB)	Show information about the WAN bridge
route -n	Kernel IP routing table Destination Gateway Genmask Flags Metric Ref Use Iface 10.223.0.0 0.0.0.0 255.255.0.0 U 0 0 0 br-wan 192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1 192.168.153.0 0.0.0.0 255.255.255.0 U 0 0 0 br-wan	Show the routing table
netstat -tuapn	… tcp 0 0 10.223.187.9:22 10.223.0.90:65297 ESTABLISHED 26182/dropbear …	Show socket connection information

arp	IP address HW type Flags HW address Mask Device 10.223.0.90 0x1 0x2 b0:25:aa:2d:d3:8e * br-wan 10.223.187.6 0x1 0x0 00:00:00:00:00:00 * br-wan	Show the address resolution protocol table.
ip a	… 5: br-wan: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000 link/ether 00:30:1a:4e:aa:09 brd ff:ff:ff:ff:ff:ff inet 10.223.187.9/16 brd 10.223.255.255 scope global br-wan valid_lft forever preferred_lft forever inet 192.168.153.1/24 brd 192.168.153.255 scope global br-wan valid_lft forever preferred_lft forever inet6 fe80::230:1aff:fe4e:aa09/64 scope link valid_lft forever preferred_lft forever …	Show information about IP addresses
fw3 print	… iptables -t filter -A zone_wan_input -p udp -m udp --dport 2000 -m comment --comment "!fw3: Allow-Socat" -j ACCEPT iptables -t filter -A zone_wan_input -p tcp -m tcp --dport 2000 -m comment --comment "!fw3: Allow-Socat" -j ACCEPT …	fw3 is a front end to iptables and can be used to configure the firewall.
cat /proc/net/nf_conntrack	… ipv4 2 tcp 6 7439 ESTABLISHED src=10.223.0.90 dst=10.223.187.9 sport=65297 dport=22 packets=767 bytes=54829 src=10.223.187.9 dst=10.223.0.90 sport=22 dport=65297 packets=640 bytes=107146 [ASSURED] mark=0 use=2 …	See exiting network connections
bmon -b **OR** **bmon -b -p wlan0**	RUN APPLICATION	Network usage information

System Information

Command	Example Output	Purpose
dmesg	… [39.776169] batman_adv: bat0: IGMP Querier appeared [39.776180] batman_adv: bat0: MLD Querier appeared [40.781438] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready [41.215540] batman_adv: bat0: Adding interface: wlan0 [41.215567] batman_adv: bat0: Interface activated: wlan0	See kernel messages
cat /var/log/messages **OR** **logread**	… Sat Feb 27 19:25:19 2021 daemon.warn dnsmasq-dhcp[2133]: no address range available for DHCP request via br-wan …	See logger messages
top	RUN APPLICATION	Check processor load
free	total used free shared buff/cache available Mem: 59436 19244 19492 952 20700 18288	Check memory usage