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

© 2020 Doodle Labs. All rights reserved.

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.

© 2020 Doodle Labs. All rights reserved.

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
```

© 2020 Doodle Labs. All rights reserved.

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.com/wp-content/uploads/json-rpc-API.zip

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.

© 2020 Doodle Labs. All rights reserved.

```
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.

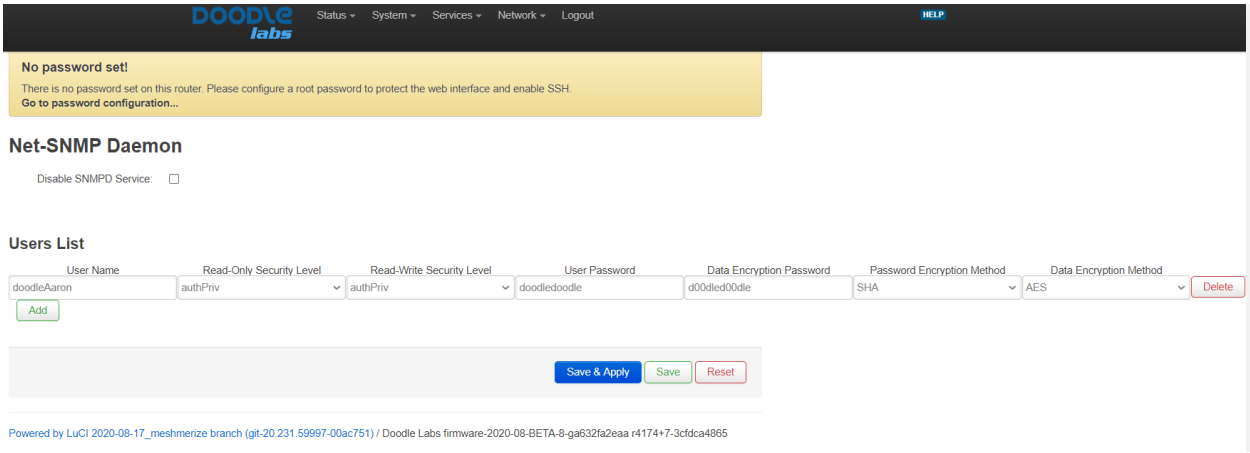
```
curl -k https://<IP-ADDRESS>/ubus -d '
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "call",
  "params": [ "<TOKEN>", "file", "read", { "path": "/etc/board.json" } ]

}'
```

© 2020 Doodle Labs. All rights reserved.

Appendix C – Using SNMP

In order to enable SNMP, navigate to services→SNMPD in the web GUI. Uncheck Disable SNMPD Service, and fill in the table. Finally click Save & Apply.



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

```
#!/bin/sh
IPADDR="10.223.255.46"
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. WIFI-MIB.txt is shown below, and more listings are available here:

https://www.dropbox.com/sh/irl7el68b05i10p/AADwJvyguPvMIMvXeIpuA-5wa?dl=0

© 2020 Doodle Labs. All rights reserved.

```
WIFI-MIB DEFINITIONS ::= BEGIN

--
-- MIB objects for WiFi statistics
--

IMPORTS
   MODULE-IDENTITY, OBJECT-TYPE, Integer32     FROM SNMPv2-SMI
;

wifiStats MODULE-IDENTITY
   LAST-UPDATED "202005111200Z"
   ORGANIZATION "www.doodlelabs.com"
   CONTACT-INFO
     "email:    tech_support@doodlelabs.com"
   DESCRIPTION
         "MIB objects for WiFi statistics"
   REVISION     "202005111200Z"
   DESCRIPTION
         "First draft"
   ::= { 1 3 6 1 3 12314 }


--
-- top level structure
--

wifiIFsTable      OBJECT IDENTIFIER ::= { wifiStats 1 }
connTable         OBJECT IDENTIFIER ::= { wifiStats 2 }



--
-- wifiIFsTable structure
--
ifaceName  OBJECT-TYPE
   SYNTAX      String
   MAX-ACCESS  read-write
   STATUS      current
   DESCRIPTION
   "Name of the WIFI interface"
   DEFVAL { NaN }
   ::= { wifiIFsTable 1 1 }
```

© 2020 Doodle Labs. All rights reserved.

```
ESSID OBJECT-TYPE
   SYNTAX      String
   MAX-ACCESS  read-write
   STATUS      current
   DESCRIPTION
   "ESSID of the WIFI interface"
   DEFVAL { NaN }
   ::= { wifiIFsTable 1 2 }

BSSID OBJECT-TYPE
   SYNTAX      String
   MAX-ACCESS  read-write
   STATUS      current
   DESCRIPTION
   "BSSID of the WIFI interface"
   DEFVAL { NaN }
   ::= { wifiIFsTable 1 3 }

mode OBJECT-TYPE
   SYNTAX      String
   MAX-ACCESS  read-write
   STATUS      current
   DESCRIPTION
   "Mode of the WIFI interface"
   DEFVAL { NaN }
   ::= { wifiIFsTable 1 4 }

txpower OBJECT-TYPE
   SYNTAX      Integer32
   MAX-ACCESS  read-write
   STATUS      current
   DESCRIPTION
   "TX power of the WIFI interface in dBm"
   DEFVAL { NaN }
   ::= { wifiIFsTable 1 5 }

signal OBJECT-TYPE
   SYNTAX      Integer32
   MAX-ACCESS  read-write
   STATUS      current
   DESCRIPTION
   "Signal of the WIFI interface in dBm"
```

© 2020 Doodle Labs. All rights reserved.

```
    DEFVAL { NaN }
    ::= { wifiIFsTable 1 6 }

noise OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Noise of the WIFI interface in dBm"
    DEFVAL { NaN }
    ::= { wifiIFsTable 1 7 }

channel OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Channel of the WIFI interface"
    DEFVAL { NaN }
    ::= { wifiIFsTable 1 8 }

frequency OBJECT-TYPE
    SYNTAX      String
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Frequency of the WIFI interface in GHz"
    DEFVAL { NaN }
    ::= { wifiIFsTable 1 9 }

bandwidth OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Bandwidth of the WIFI interface in MHz"
    DEFVAL { NaN }
    ::= { wifiIFsTable 1 10 }

connections OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
```

© 2020 Doodle Labs. All rights reserved.

```
    "Amount of devices connected to the WIFI interface"
    DEFVAL { NaN }
    ::= { wifiIFsTable 1 11 }



--
-- connections structure
--
ifaceName OBJECT-TYPE
    SYNTAX      String
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Name of the interface from which the connection statistics are displayed"
    DEFVAL { NaN }
    ::= { connTable 1 1 }


deviceMAC OBJECT-TYPE
    SYNTAX      String
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "MAC address of the connected device"
    DEFVAL { NaN }
    ::= { connTable 1 2 }


signal OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Signal value of the device in dBm"
    DEFVAL { NaN }
    ::= { connTable 1 3 }

signalAvg OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Average signal value of the device in dBm"
```

© 2020 Doodle Labs. All rights reserved.

```
    DEFVAL { NaN }
    ::= { connTable 1 4 }

inactiveTime OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Inactive time of the device in ms"
    DEFVAL { NaN }
    ::= { connTable 1 5 }

connectedTime OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Connected time of the device in s"
    DEFVAL { NaN }
    ::= { connTable 1 6 }

rxBytes OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Rx bytes of the device"
    DEFVAL { NaN }
    ::= { connTable 1 7 }

rxPackets OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Rx packets of the device"
    DEFVAL { NaN }
    ::= { connTable 1 8 }

txBytes OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
```

© 2020 Doodle Labs. All rights reserved.

```
    "Tx bytes of the device"
    DEFVAL { NaN }
    ::= { connTable 1 9 }

txPackets OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Tx packets of the device"
    DEFVAL { NaN }
    ::= { connTable 1 10 }

txRetries OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Tx retries"
    DEFVAL { NaN }
    ::= { connTable 1 11 }

txFailed OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Tx fails"
    DEFVAL { NaN }
    ::= { connTable 1 12 }

rxDropMisc OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Rx drop misc"
    DEFVAL { NaN }
    ::= { connTable 1 13 }

txBitrate OBJECT-TYPE
    SYNTAX      String
    MAX-ACCESS  read-write
    STATUS      current
```

© 2020 Doodle Labs. All rights reserved.

```
    DESCRIPTION
    "Tx bitrate of the device"
    DEFVAL { NaN }
    ::= { connTable 1 14 }

rxBitrate OBJECT-TYPE
    SYNTAX      String
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Rx bitrate of the device"
    DEFVAL { NaN }
    ::= { connTable 1 15 }


END
```

© 2020 Doodle Labs. All rights reserved.