

USERS GUIDE

SNAP Sniffer



Wireless Technology to Control and Monitor Anything from Anywhere™

© 2009 Synapse Wireless, Inc., All Rights Reserved.
All Synapse products are patent pending.
Synapse, the Synapse logo, SNAP, and Portal are all registered trademarks of Synapse Wireless
132 Export Circle
Huntsville, Alabama 35806
877-982-7888

Doc# 600026-01A

Introduction 4

Installation 4

Using the SNAP Sniffer 5

Important Notes..... 9

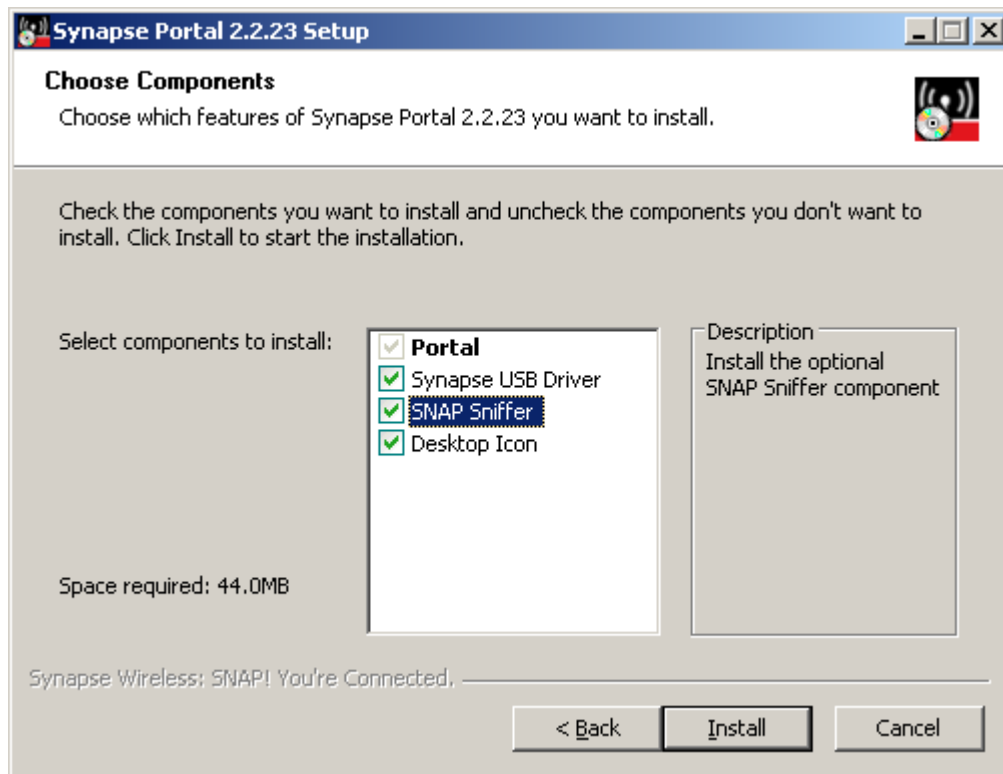
Introduction

The SNAP Sniffer provides developers more insight into how their SNAP network works and aids in the debugging of SNAPpy scripts. Specifically the SNAP Sniffer shows detailed information about packets being sent by the nodes in a SNAP network.

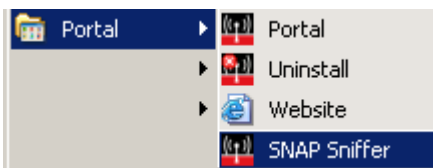
The SNAP Sniffer is comprised of two parts, the GUI which displays packets, and a special firmware image for a SNAP node which sends received packets to the GUI.

Installation

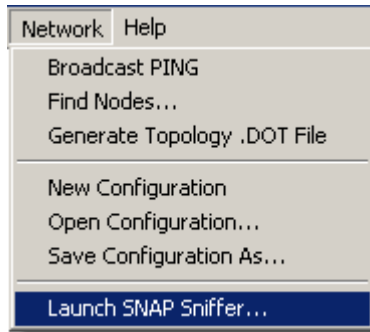
The files needed to run the SNAP Sniffer are included with Portal starting with version 2.2.23. By default the SNAP Sniffer is selected to be installed:



Once installed the SNAP Sniffer can be started from the Start Menu:



Also the SNAP Sniffer can be started from the Network menu with in Portal:

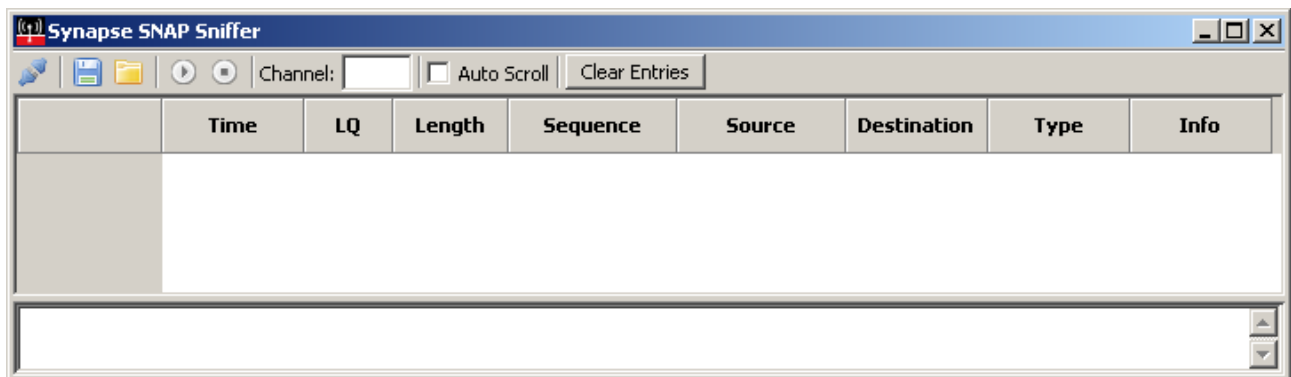


Before starting the SNAP Sniffer, you will need to convert one of your SNAP nodes to a sniffer node. This node must be directly connected to the PC's serial or USB port while sniffing. Portal can be used to load the special "SNAP Sniffer" firmware image onto the node.




Along with installing the GUI portion of the SNAP Sniffer, a series of Synapse Firmware Image (.SFI) files are installed. The .SFI files are used to load the SNAP Sniffer image on to a node. The process for loading the .SFI onto a node is the same as upgrading the firmware on any SNAP node. Please see the "SNAP Reference Manual" or "Portal Users Guide" for detailed instructions on this process. The only SNAP node that needs to be running the Sniffer firmware image is the node directly connected to the PC that is running the SNAP Sniffer GUI.



Using the SNAP Sniffer

Once launched, the SNAP Sniffers main GUI window should be displayed:

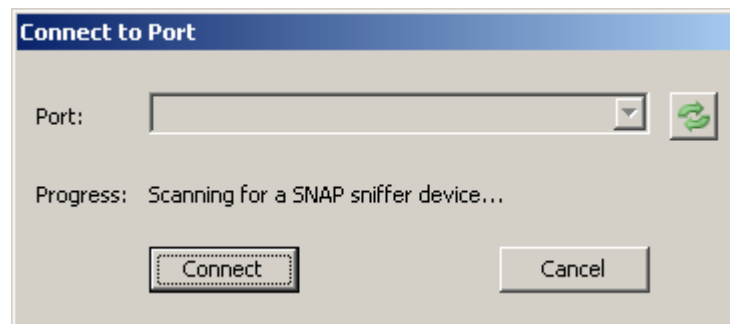


The toolbar allows quick access to the SNAP Sniffer's main functions:

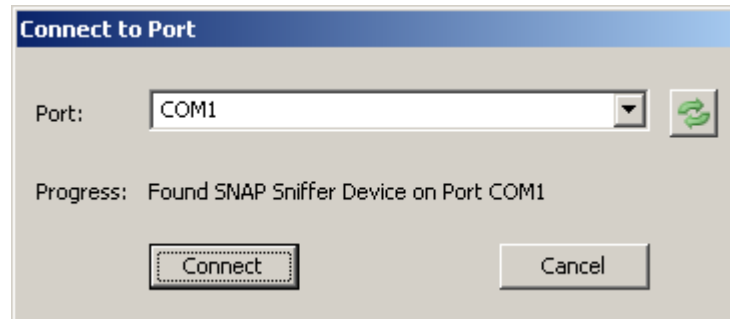
Toolbar Option	Description
	Connect/disconnect from a SNAP Sniffer node
	Save the currently captured packets to a Synapse SNAP Sniffer (.sss) file
	Opens a .sss file of previously captured packets. This option is only available when disconnected from a SNAP Sniffer node

	Instruct the SNAP Sniffer node to start capturing packets from the selected channel
	Instruct the SNAP Sniffer node to stop capturing packets
Channel: <input type="text"/>	Select the channel to have the SNAP Sniffer node to listen on
<input type="checkbox"/> Auto Scroll	Continuously scroll to display the last received packet
<input type="button" value="Clear Entries"/>	Clear all received packets from the capture buffer


When connecting to a SNAP Sniffer node, the familiar connect dialog is displayed:



Once a SNAP sniffer device has been found it will indicate which port the device is located on:



Click the “Connect” button if the correct sniffer was found, or the “Port” drop down can be used to select the correct COM port.

Now that the GUI is connected it is possible to start receiving packets from a channel. For example, to see the SNAP packets being sent on channel 11, type 11 in the channel box and press the  button. As packets are received they are displayed in the grid below the toolbar:

	Time	LQ	Length	Sequence	Source	Destination	Type	Info
1	0.000	36	21	af	00000b (002fa1)	0001 TTL=04	Multicast RPC	Method: vmStat(5, 3)
2	0.014	21	21	af	00000b (00138b)	0001 TTL=03	Multicast RPC	Method: vmStat(5, 3)
3	1.882	20	22	87	00138b	0001	Mesh Broadcast	Type: RREQ - 00138b is looking for 00000b (MAXHOPS=2)
4	1.881	65	25	f3	0035d2	0001	Mesh Broadcast	Type: RREQ - 00138b is looking for 00000b (MAXHOPS=1)
5	1.899	36	22	b1	002fa1	00138b	Mesh PTP	Type: RREP - 00000b responding to 00138b
6	1.908	20	8	b1	00138b	002fa1	ACK	
7	1.911	22	41	88	00138b	00000b (002fa1)	RPC	Method: tellVmStat(5125, 'blank')
8	1.908	35	8	88	002fa1	00138b	ACK	
9	2.665	63	22	f4	0035d2	0001	Mesh Broadcast	Type: RREQ - 0035d2 is looking for 00000b (MAXHOPS=2)
10	2.663	20	25	89	00138b	0001	Mesh Broadcast	Type: RREQ - 0035d2 is looking for 00000b (MAXHOPS=1)
11	2.666	35	25	b2	002fa1	0001	Mesh Broadcast	Type: RREQ - 0035d2 is looking for 00000b (MAXHOPS=1)
12	2.695	35	22	b3	002fa1	0035d2	Mesh PTP	Type: RREP - 00000b responding to 0035d2
13	2.704	64	8	b3	0035d2	002fa1	ACK	
14	2.720	64	43	f5	0035d2	00000b (002fa1)	RPC	Method: tellVmStat(9477, 'Lighter')
15	2.720	37	8	f5	002fa1	0035d2	ACK	

In this example, Portal was started which caused us to send a “broadcast ping” and discover the two other nodes in this network.

The grid contains the following pieces of information per packet:

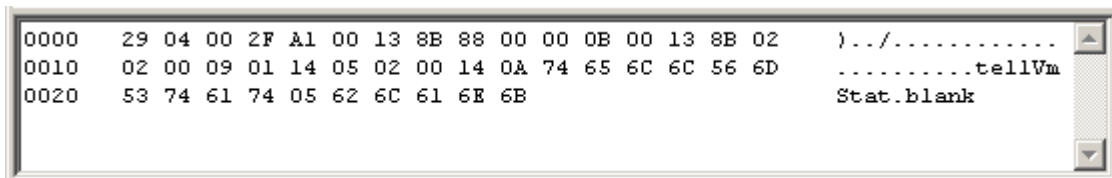
Column Name	Description
Time	The time, in seconds, of packet receipt as calculated by the SNAP Sniffer node
LQ	The link quality as received by the SNAP Sniffer node in negative dBm (lower is stronger)
Length	The total length in bytes of the received packet
Sequence	The sequence number, as determined by the sender, of the received packet
Source	The originating source address of the received packet. An address in parenthesis indicates the address of the node who forwarded the packet
Destination	The final destination address of the received packet. An address in parenthesis indicates the address of the node who forwarded the packet
Type	The packet type
Info	Information about the received packet which varies by packet type

Taking a look again at the example capture from Portal starting up we can determine the following about each packet:

1. Portal at address 00.00.0b sends a multicast RPC vmStat with parameters 5 and 3 which are forwarded by the bridge node, 00.2f.a1

2. The vmStat(5, 3), which is the “broadcast ping”, was re-broadcasted by node 00.13.8b
3. Node 00.13.8b sends a mesh broadcast route request (RREQ) message trying to determine a route to Portal at address 00.00.0b. Note that the node waited 1.882 seconds to respond to the vmStat since Portal asked all nodes to randomize their response over a 3 second interval.
4. Node 00.35.d2 re-broadcasted the RREQ packet
5. Portal responds back with a mesh point-to-point route reply (RREP) message to node 00.13.8b via the bridge node, 00.2f.a1
6. Node 00.13.8b ACKs the RREP packet
7. Now that node 00.13.8b knows the route to Portal, he sends an RPC call to Portal, invoking the tellVmStat function
8. Since the sequence numbers match, we see that the previous packet was ACK’d
9. Node 00.35.d2 now tries to determine a route to Portal
10. A re-broadcast of the previous packet
11. Another re-broadcast of the RREQ
12. The RREP from Portal via the bridge node, 00.2f.a1
13. ACK of the RREP
14. Node 00.35.d2 calling Portal’s tellVmStat with the requested information
15. The final packet is the ACK indicating the successful reception of the tellVmStat

The SNAP Sniffer can also display a raw hexadecimal display of each packet received, by clicking on the packet in the grid:



Important Notes

Currently the SNAP Sniffer does not receive or display any information about AES-128 encrypted SNAP packets. If the SNAP Sniffer node receives an encrypted packet it does not transmit it to the GUI. Additionally, if a packet is not a valid SNAP packet it will not be decoded by the GUI.

A SNAP Sniffer node cannot be used for any function other than capturing received packets to the SNAP Sniffer GUI. This prevents the SNAP Sniffer node from acting as a repeater or bridge. However, it is possible to reload the regular SNAP firmware on the node. Once the regular SNAP firmware is reloaded on the node, it can resume running SNAPpy scripts, etc.

It is also important to note that the sniffer can only show packets that occurred within its wireless reception range. If you have a large (mutli-hop) network, there may be remote communications taking place that the SNAP sniffer cannot “hear”.