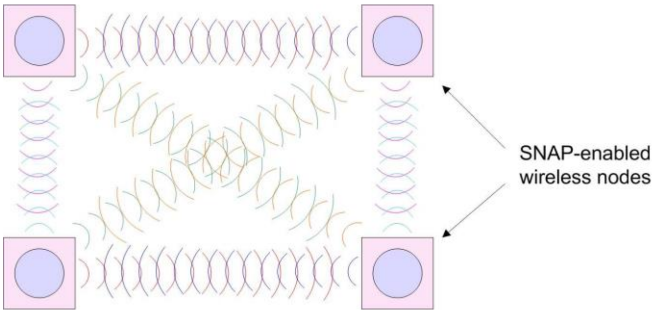
AMEREX – SNAP Network Operating System

Technical Description

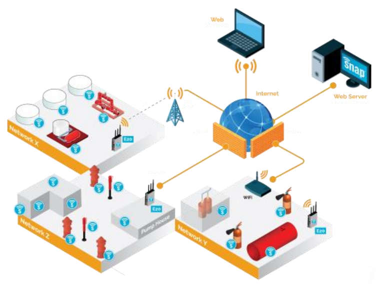
The SNAP network operating system is the protocol spoken by all Synapse Wireless devices. This networking protocol can run on any of AMEREX Integrated nodes.
With the AMEREX Integrated protocol firmware installed, the device automatically forms an ad-hoc radio mesh network with other SNAP devices in range, so each can pass information back and forth, and can relay messages to other SNAP devices that might be out of the original sender's range.
Each of these devices is a combination of a data radio and a microprocessor (literally, "small computer"). Each one can not only relay, send, and receive instructions from somewhere else, it can apply some of your own business intelligence using SNAPpy, an easy scripting language based on the powerful, popular Python programming language.
It is easy to write your own scripts to monitor input signals (analogue or digital) and control outputs. You can send and receive serial data, or connect to other devices.
With your own SNAPpy script in place, the device can be smart enough to know what signals and messages, or what combination of signals and messages, is important enough to act upon – whether that action is setting some output locally, or sending a message to another device in the network.
This means you can use SNAP products to keep track of and control what's going on somewhere else.
The control can come from explicit commands you send, or from automated instructions triggered in response to timed events or changes to the environment.

SNAP devices are social entities, and when you power them up they are happy to find other devices with which they can communicate. By default, they are configured for automatic mesh routing, which means they not only talk to each other as peers, they also act as go-betweens, relaying messages between peers who might not be able to hear each other.



Four SNAP devices in a Mesh Network

In other words, you don't need to do anything special to start up a network of devices that can monitor and respond to each other. With a line-of-sight range up to three miles, your devices can be distributed over a wide geographical area (or all grouped in the same building) and still maintain automatic communication between devices.
The term SNAP has also evolved over time to refer generically to the entire product line. For example, we often speak of "SNAP Networks," "SNAP Nodes," and "SNAP



Applications." With just a little bit of background, it's easy to see how all these parts fit together.

AMEREX—Technical

The Pieces and Parts

Any SNAP network is going to have some collection of SNAP devices associated with it. In order to best describe these parts and how they work together, it may be best to start by defining some terminology.

Nomenclature

With all the various SNAP elements in the conversation, it is easier to keep up with what's going on if you know the difference between a SNAP widget and a SNAP doodad (so to speak). This list will introduce you to the most common terms and explain how they fit together.

SNAP

SNAP, as mentioned before, is the protocol used by all SNAP devices. It is the underlying architecture that allows devices to talk to each other, comprising the communication and control structures between SNAP devices. SNAP is the infrastructure that creates the mesh network.
Most SNAP communications come in the form of a request by one SNAP device for another SNAP device to do something. At the most simple level, that request can be "Here, take (and process) this data." It can also be more elaborate, such as "Take a look at the thermal input you have from the device to which you're connected and let me know if I need to warn somebody of a pending meltdown."
SNAP also supports a "transparent data mode," where data coming into a device passes directly through to another SNAP device without being examined or acted upon.

SNAP Device

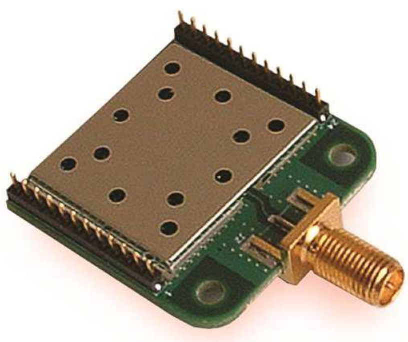
SNAP devices, then, are devices that are running the SNAP protocol so they can communicate with other SNAP devices. Each SNAP device includes a microprocessor of some sort and a communication interface. While the communication interface is typically a radio, it can also be a serial port, Ethernet connection, etc. It can even be several of these.
SNAP devices include SNAP Engines built by Synapse Wireless (or other vendors) and devices that other companies build into their own products that include the SNAP protocol.

SNAP Node

A SNAP Node is a little bit more of an abstraction than a SNAP device, though on some level the two directly correspond to each other. Each device in a SNAP network is referred to as a node in that network. In general, the term *SNAP device* will be used to refer to the product or tool that includes SNAP, and *SNAP Node* will be used in the context of networking.
Each SNAP Node necessarily is a SNAP device, and each SNAP device, when included in a network, is a SNAP Node.

SNAP Engine

A SNAP Engine is a SNAP device that matches a particular footprint and form factor in order to provide a common interface for development. SNAP Engines are produced (for various CPU/Radio platforms) by Synapse Wireless and by a few other hardware developers. SNAP Engines provide an easy way to test various hardware for prototyping, proof-of-concept work, and small-scale implementations of final products. For larger-scale final implementations, it may be more appropriate to develop custom hardware and load it with SNAP, depending on the requirements.



AMEREX—Technical

Each hardware combination has its own strengths, so you cannot necessarily drop one variety of SNAP Engine into an environment customized for a different SNAP Engine without some adjustments. But within the scope of a particular engine (such as the RF100), one SNAP Engine can be replaced by another seamlessly.

Bridge Node

A bridge node is a SNAP Node used to bridge the connection between one network section and another. For example, a SNAP device connected to your PC, using either a serial port or a USB port, forms the bridge between other nearby SNAP devices in your wireless network, and Portal (or a custom application connecting through SNAP Connect). Similarly, a SNAP Connect site controller unit bridges a local radio network to another SNAP device or network across the Internet.

Products

AMEREX Integrated offers many products that fill the various needs of a SNAP Network. The software and hardware solutions offer the flexibility to quickly develop complicated systems to control and monitor nearly anything. You will encounter references to these products in the various AMEREX Integrated literature. Note that it is tremendously unlikely that any given network will have all these hardware and software variations in it. Most networks will have a collection of nodes based on one particular platform (or SNAP Engine), and one PC-based control or monitoring point (based on Portal or SNAP Connect).

Software

Portal

Portal is a SNAP implementation to turn your PC into a SNAP device, so it can communicate with any other device (node) in your network. It is a GUI application that you run on your computer (Windows, Macintosh, or Ubuntu Linux). Using a serial connection to a bridge node, Portal provides the nexus into the rest of your network, with an easy-to-use interface.

Portal is also what you use to administrate and maintain your nodes, by loading controlling scripts into them and monitoring their communications. See the Portal Reference Manual for a complete understanding of Portal's role in your networks.

SNAP Connect

SNAP Connect is a stand-alone application you can use to allow your own client applications to access your SNAP network the same way Portal does. It establishes an XML-RPC server interface that you can reference from Java, C++, C#, Python, or nearly any other modern programming language – as well as a great many older languages. Through this SNAP Connect interface you can send instructions and data to and receive instructions and data from any other node in your network.

SNAP Sniffer

The Portal installation includes firmware images that allow you to convert a SNAP Node into a SNAP Sniffer. As a sniffer, the device no longer interacts with other nodes in your network. But combined with Sniffer software running on your PC, it reports back all the network traffic it hears (on a specified network channel). This can be valuable for troubleshooting communications within a network.

SNAP Firmware

The Synapse Wireless SNAP firmware is the code that turns a piece of hardware into a SNAP device. Without the firmware the radio and microprocessor may have the ability to communicate, but have no instructions on how to do so. In general, firmware provides these instructions, and SNAP firmware tells the devices how to participate in a SNAP network, unifying communications and control across disparate physical layers and across different platforms.
If you acquire SNAP hardware, the firmware will already be loaded. If you build your own hardware and want to include SNAP, you will need to acquire licenses for the SNAP firmware, and will need to load that firmware into your devices before they can communicate with other SNAP devices.

AMEREX—Technical

Each hardware platform has its own firmware build, and most platforms have several firmware builds available for them. The normal build includes all the usual SNAP features.
For testing purposes, some users like to have access to a “debug” build of the firmware, which provides more details about exceptions caught in users’ code. The debug build runs slightly slower than the normal build and provides a little less code space for user applications, so most users would not run this in a production environment.
Most platforms also have an AES-128-capable build available. This build allows for much stronger encryption than the Basic Encryption available in the normal firmware builds. The AES-128 firmware is not available in all jurisdictions.
SNAP devices from Synapse Wireless will ship with the most current release of the firmware installed.
Synapse Wireless continually updates the software to include new features, so users may want to update their firmware at times. The process is simple, and is covered in the Portal Reference Manual.

Hardware

SNAP Engines
Synapse manufactures a variety of SNAP Engines based on various hardware platforms. There are several offerings in the 2.4 GHz spectrum, and other options in the sub-GHz range.

USB Bridge Nodes
Synapse has several USB devices available to bridge between network sections. SNAP Sticks, about the size and shape of a USB “thumb” drive, connect your 2.4 GHz wireless network to Portal or, through SNAP Connect, to your own custom application. There are also SN132 USB boards that accept any SNAP Engine and plug directly into a USB jack, for the same purpose.

Site Controller
The site controller is a SNAP-enabled embedded connectivity appliance that allows you to connect between disparate SNAP networks over TCP/IP networks, such as the Internet. It provides a Linux environment on which you can deploy your own application to control and monitor your SNAP network, coupled with a SNAP Engine to tie the Linux environment to an existing SNAP wireless network. In this way, a SNAP network in one location can control and respond to other Internet connected SNAP nodes anywhere in the world. The applications you install in the site controller can also perform any other function you care to program in response to feedback from your widely distributed SNAP network.

SNAP Link
SNAP Link modules provide a simple way to replace wired serial connections between two devices that need to communicate with each other. When you have two devices that require a serial connection (RS232, RS422, or RS485) and connecting them with a physical wire is impossible or impractical, SNAP Link modules can wirelessly replace the cable. Because SNAP Link modules are SNAP Nodes they automatically take advantage of mesh routing, allowing you to extend your effective range and increase reliability.

What is a mesh?

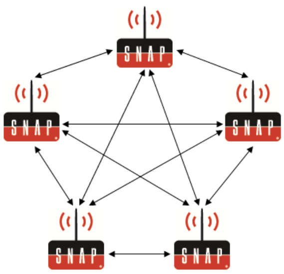
Automatic mesh routing is one of the features that makes SNAP networks so useful. As soon as you power up a SNAP node, it automatically becomes a peer in whatever SNAP network you have available. There is an important detail here: All nodes in a SNAP network are peers. Each radio node can talk directly to any other node within radio range, and indirectly to any other node in the network.

AMEREX—Technical

There is no need for a “master” or “coordinator” in a SNAP network. Though it’s easy to think of the Portal node (or SNAP Connect node) as “the boss” of the network (and in terms of command/control it may serve that function), the underlying communication structure does not require – or even accommodate – that any one node act as a hub or traffic director.
Every node in the network does its part to make sure all other nodes receive the messages intended for them, regardless of how the messages originate. By default, nodes forward messages addressed to other nodes that are out of radio range of their source, and nodes retransmit multicast messages, to ensure that everyone within range gets a chance to hear them.

Simple, Short-Range Group

In a network topology where all nodes in the environment will always be within radio range of each other, you can improve response time by telling each node to not bother sending periodic route requests to make sure it knows a path to its peers. Unicast messages will always go out with the expectation that the destination will be in range, and routing requests (which make use of mesh routing to have nodes forward messages to other nodes) will only happen if the initial transmission fails.



Very Dense Multicast Cluster

If you have many nodes within range of each other and you need to communicate to them by multicast, your messages (or their responses to your messages) can get lost in the noise generated when all the nodes start rebroadcasting your message to make sure everyone else heard it.
(Imagine how hard it would be to have a conversation in a crowded room if every time you said something, everybody else repeated it for you.)
The diagram at below demonstrates this environment, where every node repeats every message to ensure it is heard.
Depending on the specifics of your environment, there are several ways you might configure your nodes.