UNIVERSITY OF PADOVA

SyNaPse 0.9

User's manual

10/14/2009

SyNaPse v 1.0 – User's Manual

Introduction

SyNaPse is a tool that enables reliable over-the-air reprogramming for a multi-hop sensor network.

Using SyNaPse it is possible to distribute new applications or upgrades to all the nodes in a Wireless Sensor Network by accessing only one node (the BaseStation), which forwards the new code to the rest of the network. Compared to similar approaches, the major contribution of SyNaPse is that, using the FEC (Forward Error Correction) technique known as Fountain Codes to recover from transmission errors, its performance does not suffer a number of critical situations such as the feedback implosion problem.

SyNaPse has been developed by the researchers of the SigNET lab at the University Of Padova. It is released under the BSD license, as reported at the end of this document.

SyNaPse is a collection of three main components:

- A bootloader, which sits in front of every application running on the nodes and manages the reprogramming operations.
- A GoldenImage, the main part of SyNaPse. The GoldenImage is a TinyOS application that is installed in every node and must be loaded by the bootloader when the over-the-air functionalities are required.
- The TinyOS libraries that can be used, but don't have to be used, from applications to interface with the bootloader. Application designers can decide to create applications that are completely unaware of SyNaPse and its bootloader. How to switch to the GoldenImage in this case will be explained later in this document.

This manual does not provide technical details on the design and implementation of SyNaPse. For more information concerning the protocol please refer to the publication list at the end of this document.

SyNaPse Installation

System Requirement

Synapse is currently developed for the **Telosb** platform. It has been tested successfully with Telosb and TmoteSky sensor nodes.

SyNaPse to work properly your system has to be configured with TinyOS 2.x. The current distribution of SyNaPse is tested with TinyOS-2.1.0. TinyOS is only required to compile and install the SyNaPse main application and manage it. The applications that the user wants to distribute and run in the network can come from any source and can be written in any programming language. All it is required for distributing and running them is their binary code in *ihex* format. The utility `msp430-objcopy` provides an efficient toolkit to convert binaries from different format to the *ihex* format. More information can be found in the *mspgcc* toolkit.

In addition to TinyOS and its standard tools SyNaPse requires a Python interpreter (SyNaPse has been tested with version 2.4 and 2.6), and the command line utility `xmlstarlet` to be installed on your PC.

Patching the TinyOS tree

SyNaPse installs all the source code and libraries in the `SynapseNetProg` folder in the `contrib` subtree of the TinyOS distribution. It has also to modify the following files in the `support/make` directory:

- `support/make/telosb.target`
- `support/make/msp/install.extra`
- `support/make/msp/reinstall.extra`
- `support/make/Synapse.extra`

If your system is fully setup to work with TinyOS (i.e., the paths to $TOSDIR, $TOSROOT and $MAKERULES are pointing to the correct locations) all you need to do is to extract the files from the SyNaPse archive and launch the install script `install.sh`. The utility will patch your make files and import SyNaPse in the TinyOS tree.

After installing SyNaPse two new variables have to be added to your environment. You can do it by editing the `.bashrc` file in your home folder, or the file that sets the TinyOS environment in your system. Add the following lines:

```
export TINYOS_NP=Synapse

export PATH+=:${TOSROOT}/contrib/SignetNetProg/utils/
```

Your system is now ready to use SyNaPse.

SyNaPse Usage

Installing the GoldenImage on nodes

Preparing all the nodes for using Synapse is very easy. All you have to do is enter the folder `GoldenImage` in the `$TOSROOT/contrib/SynapseNetProg` and then simply use the command:

```
make telosb install.<nodeID> Synapse,none
```

It is very important to verify that the symbol LOADABLE is not defined in the `Makefile`. Moreover, it is preferable to assign to every node a different ID. Nodes will keep this information safe in their memory to have it available for every application that will be distributed using SyNaPse. The use of the symbol LOADABLE will be explained later in the *"Advanced Usage"* section in this manual.

Every node can be connected through the USB and can act as a BaseStation. Throughout this manual we use the term BaseStation to indicate the node that is connected to a PC, from which it receives new applications to be disseminated and other control commands. There's no privileged node in the network, every SyNaPse node can be connected to a USB port at any time and assume the role of BaseStation, as the GoldenImage is the same for all nodes.

Formatting the External Flash

SyNaPse uses the external flash memory to store the binaries of all the different application that are sent and received over time. The space in this volume is limited and can contain a variable number of applications, depending on their size.

After installing the GoldenImage in the nodes it is recommended to format this storage. All the commands to the BaseStation as well as all the other nodes are managed through the utility `suino`, located in the `SynapseNetProg/utils` folder. To format the memory in the BaseStation, simply run:

```
suino -c <comport> formatBS
```

where `<comport>` is the address of the serial device the node is connected to. After executing this command the node will reboot and format its memory. When the memory is erased the node creates a new partition table and saves the Golden Image in the first partition (the red LED is blinking during the process). On completion the GoldenImage is load again and the node is ready to execute new commands.

The default SyNaPse volume can store approximately 320KB of data, including the GoldenImage, and up to 18 different applications. To learn how to increase or reduce this storage space or how to use the external flash memory in your application without jeopardizing the correct behavior of SyNaPse refer to the *"Advanced Usage"* section of this manual.

As the storage space is not unlimited and the applications are not deleted by SyNaPse during its regular operation, we recommend to periodically formatting the memory to avoid running out of space. If

SyNaPse cannot allocate enough space for a new application it will ignore it. To format the external flash memory of the other nodes in the network simply use the `suino` command as follows:

```
suino -c <comport> prepare

suino -c <comport> format
```

The *prepare* command stops all the current operations in the network to avoid weird behavior during the execution of special commands such as format. The blue LED indicates that the node is in "prepared" status, when the format command is received the node reboots and the red LED blinks during the formatting operation. Also in this case the GoldenImage is saved in the first partition. When the process is completed the green LED on the sensor turns on, and the SyNaPse GoldenImage is again ready.

Injecting a new application on the BaseStation

In order to disseminate a new application in the network the user must first load it into the external flash memory of the BaseStation. To do so the first step is to compile the application using the regular TinyOS command:

```
make telosb
```

This will produce an *Intel-executable* file (main.*ihex)*, which can be sent to the node. There ihex file can come from any source, it is not mandatory that it is a TinyOS application. Any binary file in Intel format is accepted and can be successfully sent and loaded. In TinyOS the file is usually located in the *build/telosb/* directory of the source code. Be aware that this file is generic and does not set the TOS_NODE_ID symbol. To learn how to maintain the node address on each node during the reprogramming operations, please refer to the "Preserving TOS_NODE_ID" section later in this guide. Once the executable file is created to load it on the BaseStation use the following command:

```
suino -c <comport> write_app <appID> <ihex file>
```

where `<appID>` is a 16 bit hexadecimal value that will identify the application in the nodes' filesystem. If two applications are stored with the same ID only the most recent one will be accessible. The `<appID>` value must start with the identifier `0x`, followed by four hexadecimal ciphers.

During the operation the LED indicating serial activity on nodes will blink, and a progress bar on the console will notify the user of the fraction of application already sent. The process of writing on the external flash requires a constant voltage at the Flash chip. If the node is connected with a USB cable it might experience problems during this operation, especially if using a USB2.0 hub. To avoid this problem use a powered USB hub.

Disseminating a new application

Once the application is successfully loaded on the BaseStation it can be distributed to the network using the command:

```
suino -c <comport> transfer <appID>
```

to begin the dissemination process. The operation is completely automatic, and will take from few seconds up to some minutes depending on the size of the application and the number of hops that it has to travel to reach the farthest node. On completion the Synapse nodes have all the LEDs on, which indicates that they possess the whole image. Removing or resetting the nodes that are done already is not recommended until the whole network is programmed, as they might still serve as forwarders for the farther nodes.

Loading the new application on remote nodes

Once the application has reached all the nodes in the network it is possible to reboot the nodes loading the new application. To do so, once again we need first to stop the current SyNaPse activities with:

```
Suino -c <comport> prepare
```

followed by:

```
suino -c <comport> load <appID>
```

The nodes will reboot and the bootloader will load the new application from the external flash. At the next reset the node will load the GoldenImage again, to be ready for other code disseminations. To disable this safety mechanism intended to allow the nodes to recover from a faulty application, read the "*Disabling the safe reboot option of the Bootloader*" section, later in this manual.

Advanced Usage

Disseminating a GoldenImage upgrade

It is very easy to disseminate an upgrade to the GoldenImage to all nodes, if needed. The only component that cannot be overwritten without a USB connection to the node is the bootloader. The upgrade consists of a simple 5-step procedure:

1) Compile the new version of the golden image with the symbol using the following command:

   ```
   CFLAGS+=-DLOADABLE make telosb
   ```

2) Upload the application on the BaseStation choosing any unused ID:

   ```
   suino -c <comport> write_app 0x01f3 build/telosb/main.ihex
   ```

3) Disseminate the application to all nodes:

   ```
   suino -c <comport> transfer 0x01f3
   ```

4) Send the load command with the ID of the new GoldenImage

   ```
   suino -c <comport> prepare

   suino -c <comport> load 0x01f3
   ```

5) Issue the `format` command to store the new GoldenImage in the first partition of each node's memory, so that it's the one that is loaded when SyNaPse dissemination is required:

   ```
   suino -c <comport> prepare

   suino -c <comport> format
   ```

The symbol LOADABLE is required to force the GoldenImage to read the TOS_NODE_ID from the ROM instead of writing it. By default, if the LOADABLE symbol is not define, the node address will be overridden by the GoldenImage and all nodes would have the same ID after being reprogrammed, with no possibility of recovering it later.

Reading the TOS_NODE_ID

During the first installation of the bootloader and the GoldenImage in the node, through the USB port, a different TOS_NODE_ID for each sensor is selected through the `tos-set-id` tool. When the GoldenImage is loaded it saves this information on a specific location in the microcontroller ROM, so that all applications can access it. This is mandatory since the images that are received with SyNaPse are not differentiated, so all nodes would have the same TOS_NODE_ID, making it very hard to identify the single nodes and very likely breaking the routing and communication protocols.

The node ID is saved in the 4th word of the IF_SEGMENT A of the msp430 microcontroller, which is a 16 bit value mapped at the memory address 0x1008. If your application is written in TinyOS 2.x, SyNaPse provides a way to quickly include the code necessary for retrieving and setting it, otherwise the programmer can write its own code to access this value. Note that the msp430 has no option to write-protect a memory location, and that once this value is lost there's no way to recover it.

To set the TOS_NODE_ID in a TinyOS application, include this code in your application module:

```
//In the list of used interfaces add, if not present already:
interface CC2420Config;
interface Synapse_BootloaderCommunication
async command void setAmAddress(am_addr_t a);

//in the Init.init() procedure add the following lines:
am_addr_t nodeId;
nodeId=call BootloaderCommunication.readNodeId();
atomic {
    TOS_NODE_ID = nodeId;
    call setAmAddress(nodeId);
}
call CC2420Config.setShortAddr(nodeId);
call CC2420Config.sync();

//also add the following event if not present already
event void CC2420Config.syncDone(error_t error){}
```

also, in your application configuration file, add the following:

```
components CC2420ControlP, ActiveMessageAddressC;
<myappP>.CC2420Config -> CC2420ControlP;
<myappP>.setAmAddress -> ActiveMessageAddressC;

components Synapse_BootloaderCommunicationC;
<myappP>.BootloaderCommunication ->
Synapse_BootloaderCommunicationC.Synapse_BootloaderCommunication;
```

Disabling the safe reboot option of the Bootloader

By default after loading an application from the external flash through a radio command, the SyNaPse bootloader is set to load the GoldenImage at the next reboot. This is a safety policy to ensure that it is always possible to revert to the GoldenImage after some bad behavior of the application. A watchdog timer is enough to recover an application even when the radio transceiver stops working. Additionally, this approach makes it very easy to interface applications from different sources and SyNaPse. There's no need to include any component from SyNaPse in the distributed applications, the only requirement is that all the applications implement some kind of reboot policy to load the GoldenImage when needed.

To disable the automatic loading of the GoldenImage it is necessary to recompile the bootloader after commenting the line that defines the symbol PRESERVE_GOLDENIMAGE. The bootloader is located in the directory `contrib/SynapseNetProg/SynapseBootloader`. To recompile the bootloader simply enter the directory and run:

```
make Telosb
```

The new bootloader is saved in the `binaries` directory.

Reading an application from the BaseStation

In order to verify the binary code of an application before it's distributed, SyNaPse allows the user to download an application from the Flash memory to the console. To do this, simply use the command:

```
suino -c <comport> read_app <appID>
```

where `<appID>` is a 16 bit hexadecimal value that will identify the application in the nodes' filesystem. If two applications are stored with the same ID only the most recent one will be accessible. The `<appID>` value must start with the identifier `0x`, followed by four hexadecimal ciphers.

Reading the partition table from the BaseStation

If the user is not sure about the number of applications that are stored on a node or their IDs, it is possible to download and print on screen the partition table using the command:

```
suino -c <comport> get_table
```

Connecting through a SerialForwarder

If for some reason the user wants to use a Serial Forwarder to connect to the node instead of sending the packets through the RAW serial port, it is sufficient to launch the SerialForwarder and use `suino` with the parameters `-p <sfport> --port telosb.`

Varying the size of the Synapse volume

SyNaPse is not using the whole external flash for storing applications. It reserves 320KBs for itself and leaves the rest for other applications. If the user wants to change the size of the volume it only has to modify the files `SynapseNetProg/volumes-stm25p.xml`. After changing this value the bootloader has to be recompiled as explained before, by running the `make telosb` command in the `SynapseNetProg/SynapseBootloader` directory.

This change has to be made before initializing all nodes, as the bootloader can't be substituted once the network is deployed. Other applications that want to use the external flash can define new volumes in the volumes file, and then use the BlockStorage component of TinyOS, after creating a symbolic link to the `SynapseNetProg/volumes-stm25p.xml` file to their application's source directory.

Appendix A- Publications

Additional information of SyNaPse can be found in the following publications:

Rossi, M., N. Bui, G. Zanca, L. Stabellini, R. Crepaldi, and M. Zorzi, **"SYNAPSE++: Code Dissemination in Wireless Networks using Fountain Codes"**, *IEEE Transactions on Mobile Computing, In Press.*

Rossi, M., G. Zanca, L. Stabellini, R. Crepaldi, A. F. Harris, and M. Zorzi, "**SYNAPSE: A Network Reprogramming Protocol for Wireless Sensor Networks Using Fountain Codes"**, *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON '08. 5th Annual IEEE Communications Society Conference on, pp. 188-196, June, 2008.*

Crepaldi, R., A. F. Harris, M. Rossi, G. Zanca, and M. Zorzi, **"Fountain Reprogramming Protocol: a Reliable Data Dissemination Scheme for Wireless Sensor Networks Using Fountain Codes",** *the ACM Conference on Embedded Networked Sensor Systems (Sensys), 2007.*

Appendix B- BSD License