Generic Network Visualization Tool for Monitoring Adhoc Wireless Networks

Girish Revadigar[1] and Chitra Javali[2]

[1] AllGo Embedded Systems, Bangalore, India
girishrevadigar@gmail.com
[2] PES Institute of Technology, Bangalore, India
chitra.javali@gmail.com

Abstract. Adhoc sensor network consists of dense wireless network having tiny, low-cost sensor nodes. Examples include military applications, and acquiring sensing information from inhospitable locations like thick forests, active volcano regions etc. Since the devices are scattered in a complex sensor network, It is very difficult to locate each node, know about its status and the topology of the wireless network. Hence the need of developing a system to visualise, control and monitor such networks arises. This paper presents implementation issues and author's contribution to design and implement a generic framework of the 'Network Visualization tool' to monitor adhoc wireless networks. The paper also elaborates system architecture, hardware and software organizations, and integration details of the proposed system with an exemplary wireless network based on standard IEEE 802.15.4 MAC protocol.

Keywords: Adhoc Wireless Networks, IEEE 802.15.4 MAC, Network Visualization.

1 Introduction

Recent advancements in wireless networking technology has enabled us to use wireless connectivity in almost all our applications. The ease of integration, support from multiple platforms, interoperability and co-existence with other technologies have made these more popular. At the same time, as the complexity of such networked systems increases, the effort required to monitor and control such systems also increases. For critical applications, if a single node is not working, then the loss of data from the node may cause serious complications.

In this paper, we describe our design, implementation and integration details of a 'Generic Network Visualization tool' for monitoring adhoc wireless networks.The implementation is based on IEEE 802.15.4 MAC protocol using AllGo's wireless nodes on Freescale's MC1321X MCU.

Section 2 explains the system architecture, section 3 describes target subsystem representing a wireless network. In section 4, the Host subsystem part featuring a PC based application details are present. Section 5 describes the integration and testing of our Network Visualization tool. Section 6 concludes the paper.

N. Meghanathan et al. (Eds.): CCSIT 2012, Part I, LNICST 84, pp. 306–315, 2012.

2 Network Visualization Tool Architecture

The framework of Network Visualization tool is designed in a generic way such that it can be easily integrated with any type of existing wireless network protocols like IEEE 802.15.4 MAC, SMAC, ZigBee[4]/ZigBee Pro[5] etc. Figure 1 shown below describes the system architecture.

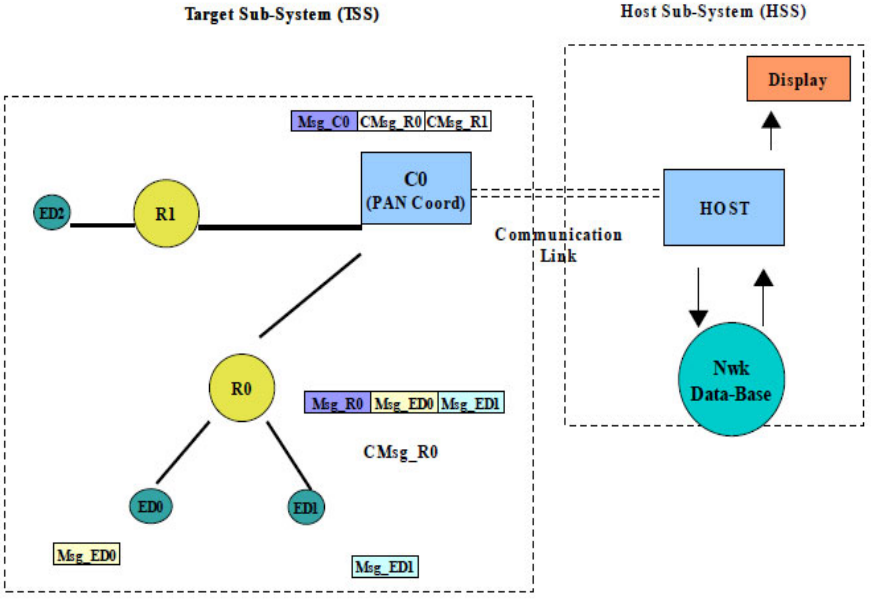

Fig. 1. Generic Network Visualization Tool Architecture

The tool consists of two mutually communicating components, a Target subsystem(TSS) and Host subsystem(HSS). Target subsystem resides on all the wireless nodes of the network viz. Coordinator, routers and end devices. The coordinator performs the task of sending network related information to the host. Host subsystem resides on the host PC/Laptop and is responsible for processing and displaying the network related information. Since the information received from target subsystem will be in a distributed form, host subsystem performs the function of analyzing and interpreting the information received from the coordinator and display it .

Since it is a visualization tool, most of the traffic will be from target subsystem to host subsystem. However a low bandwidth reverse flow is also required to configure target subsystem behavior based on user inputs provided to the host subsystem.

3 Target Subsystem (TSS)

Target subsystem is the most basic component of our visualization tool, since all the messages and information related to network are all constructed and sent to the host

subsystem for interpreting the same. The different design constraints of target subsystem are explained in detail in the following sub sections.

The target subsystem is the actual network that is being visualized. Hence the target subsystem should be least intrusive to the actual network functioning, ex. In a multi priority system, message logging by the target subsystem will be the lowest priority task. In a single priority system, logging by the target subsystem will be the task executed after the devices have completed all the pending items. A target subsystem consists of various components which are described below.

3.1 Bridge Node (TSS-BN)

This is the node in the wireless network that communicates with the host subsystem(HSS) and all the other nodes in the TSS send (over the air) the data to be logged to this particular node. It can be any node but the main network coordinator device is a preferred choice for TSS-BN. In most of the applications network coordinator should have lot of network information that has to be logged. In such a case the tool bandwidth requirements will be greatly reduced by having coordinator perform as a TSS-BN also.

3.2 TSS Message Types

The messages exchanged between TSS and HSS are classified as below

1. Information Logs (Tx): These are the logs for network visualization. Two sub-types of logging are possible - event based logging and periodic logging,
2. Debug Logs (Tx): Software debug logs
3. Configuration Msgs (Rx): Used by host to configure logging parameters (log frequency, debug logs ON/OFF, Control signals etc.)

During the initial network configuration the HSS will require information about neighbours/child devices for each device in the network. Subsequently, only 'differential configuration information' will be sent to HSS. This can be viewed as an event based logging. Event based logging is made possible by the use of TSS-Call Back (TSS-CB) functions provided by the TSS to the network application.

To display certain network attributes (average LQI, network statistics, average_ON_Time) a periodic logging is required. This implies a timer based logging of the required attribute.

To allow a limited debugging each device can log data independently that will not be interpreted by HSS. It will be displayed as it is, in a separate window and can be turned ON/OFF dynamically on user requests. (It can be used to track the progress of any individual device as if the device itself is connected to serial link).

3.3 TSS Message Structures

To facilitate all the requirements at the target side, generic message structures are designed which can be used for specific purposes, for example, the Coordinator Device can use a structure called "DevInitMsg_t" as shown below to log its initial

data to the host when it enters the TSS state machine's START_LOG state for the first time.

```
typedef struct DevInitMsg_tag
{
uint8_t devid[2];          // variable to hold the device id (short address)
uint8_t devtype;           // variable to hold the device type
uint8_t numchilddev;       // variable to hold the number of child devices.
} DevInitMsg_t;
```

3.4 Classification of TSS Messages and Functions

The messages used in TSS are classified into two types as 'Control Messages' and 'Log Messages'. Also there are two types of functions used with these message structures, viz 'APIs' and 'Callback functions'.

Control messages are those which are used by the devices in the network for the purpose of keeping track of the Host Subsystem. These can be as follows,

1. Control messages Request type - used by the end devices most of the time for querying the coordinator about the Host subsystem status
2. Control Messages Response type – used by the coordinator device to send the response back to the end devices in the network which send the query requests.

Log messages are the types of messages used by all types of devices in the network. There can be different types of log messages like the ones listed below,

1. The log message used by the coordinator and other devices in the network for logging their initial data,

```
typedef struct DevInitMsg_tag
{
        uint8_t devid[2];
        uint8_t devtype;
        uint8_t numchilddev;
} DevInitMsg_t;
```

2. End device can log its Link quality using a specific log structure as shown below.

```
typedef struct LinkQualityMsg_tag
{
        uint8_t len;
        uint8_t lqi;
} LinkQualityMsg_t;
```

3. Similar message structures can be used for different application purposes also, for example, for logging the temperature value received from a sensor etc.

```
typedef struct AppTemperatureMsg_tag
{
        uint8_t len;
        uint8_t temperature;
} AppTemperatureMsg_t;
```

3.5 TSS Messages Operations

TSS consists of all the devices in the network including Pan coordinator, and End devices, routers. Each device in the TSS has a basic state machine for their functionality but the individual functionality of the devices varies in different states based on the device type, viz Coordinator/End device or router. The basic state machine and also different functionalities of the devices in these states are explained in the subsections of this chapter.

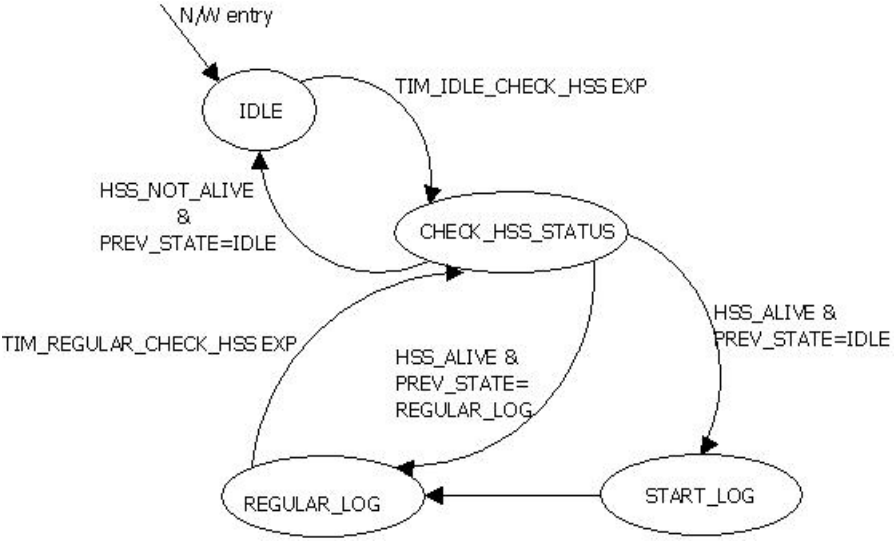

Fig. 2. TSS device state machine

The device will be in the state called "INVALID_STATE" until it joins the network, after it joins the network, the first state it enters is IDLE. The subsequent state transitions are explained as follows.

1. **State IDLE:** In this state the device will not do anything except waiting for the HSS to become active. A timer is set to expire every 10 seconds and is used to change the device tool state to a state called 'CHECK_HSS_STATUS' where the actual polling for HSS status happens. The device stays in this state until it receives 'HSS alive' response for the query of HSS status.
2. **State CHECK_HSS_STATUS:** In this state the device sends a query for HSS status and according to the present status receives one of the three

possible responses as ‘HSS alive’, ‘HSS not alive’, or ‘Refresh log’. According to the type of response obtained for the query of HSS status the device then decides its next state . If the response is ‘HSS not alive’ then the device returns to ‘IDLE’ state. If the response is ‘HSS alive’, then the device chooses its next state as ‘START_LOG’ if the initial log is not complete(ie, if the previous state from which it entered the current state was 'IDLE'.), or the device may directly enter another state called ‘REGULAR_LOG’ if the initial log is already completed.(ie, if the previous state from which it entered the current state was 'REGULAR_LOG'.)

3. **State START_LOG:** In this state the device just does its initial log where it logs all its data related to network configuration and other useful information maintained by the device.
4. **State REGULAR_LOG:** In this state the device keeps waiting for any change of event to occur in the network and if it finds one, it just logs that data. Also the periodic logging is provided by means of a timer set for a particular time (say every 5 seconds). To keep track of HSS status, a timer is set to expire at every 20 seconds, after each time the timer expires, the device state changes to ‘CHECK_HSS_STATUS’ where the actual polling for HSS status happens. If the response for the HSS status query is ‘HSS alive’ and the ‘Initial log complete bit’ in the device tool status register is set, (ie, the previous state from which it entered the current state was 'REGULAR_LOG') the device then enters ‘REGULAR_LOG’ state again.

3.6 Data Logging Mechanism

To reduce the number of air accesses and the logging latency, a buffer based logging mechanism is employed by the TSS. The information to be logged will be stored in a 'log/dump-buffer' in an appropriate message structure:DEV-SINGLE-MSG. Multiple such messages will be collected in the 'log-buf'. Then either timer-based or total-message length based (or both) logic will be employed to actually send this information over the air to the parent/coordinator. This message will be sent via another message structure:DEV-OTA-MSG. A few benefits of such a mechanism are:

1. Better control for over-the-air message lengths and frequency.
2. Better control over the logging latency.

A priority logging can be easily introduced by having a higher priority buffer that has to be flushed out first.

3.7 Timer Based Activities in TSS

All the timer based activities in TSS are specified in terms of 'TimeTicks' of a specified duration. The 'TimeTick' duration can be quite flexible if a regular processor timer is selected to provide the basic 'TimeTick'. But since an application will require the end-devices to sleep it might not be appropriate to use the regular processor timer. In such a case a basic 'TimeTick' will be equal to 'sleep duration' of the device. Thus,

to be able to efficiently handle both these cases the period based events will be tracked using the 'TimeTicks' instead of the amount of time (s or ms).

3.8 TSS Interactions with Application

TSS, a part of wireless device will be at the same level as the 'APP' layer in the application domain. Figure 3. shows the TSS interactions with the application. As mentioned earlier, TSS needs to be easily integrated onto existing application and hence the interfaces between TSS domain and Application domain needs to be well defined. TSS will need NWK APIs to send (Tx) log messages over the air, to query some typical parameters from the network data-base (Neighbour tables, routing tables etc.) or to receive (Rx) some configuration parameters/control signals over the air from coordinator.

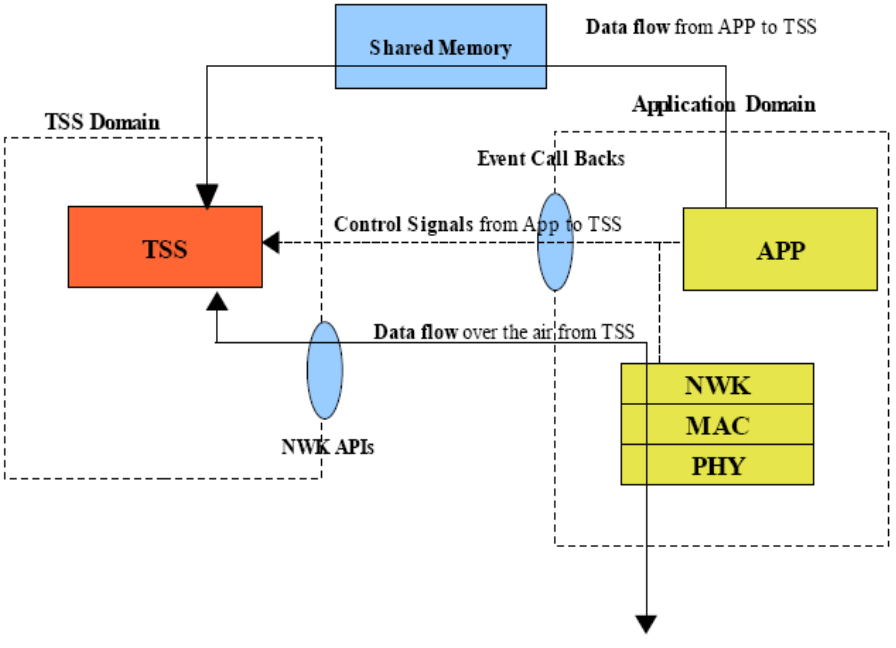

Fig. 3. TSS and application interactions

On the other hand the application might also need to interact with TSS. For ex. to inform TSS about network configuration change (association successful, addition/removal of node), register a data structure to log (actual data transfer takes place through shared memory) etc. Interactions between tool domain and application domain will be through 3 mechanisms: NWK APIs, Event Call Backs and Shared Memory.

1. **Event-CB Functions:** Call-back functions for application to inform TSS task of different events. Application will need to use these call-back functions whenever any event takes place that needs to be communicated to TSS.

2. **Network-Usage APIs:** APIs for using network/applications services. These APIs will be using the network service provided by the application and hence will need to be adapted based on the network we are integrating the TSS with.
3. **Shared Memory:** To share application data structures to be logged. To share data received over the air meant for TSS.

4 Host Subsystem

The host subsystem essentially consists of the following parts,

1. A host Pc/Laptop on which the network visualization tool runs
2. A serial port configuration module and Interpreter written in Java,
3. A database for storing the data
4. Prefuse tool kit – A Java based GUI tool (Graph visualization tool),

Steps include - Get the messages from network through Pan Coordinator using serial port, interpret the incoming messages and update the data structure which stores the current graph.

5 Integration and Testing

We have tested the tool by integrating with a simple star network based on standard IEEE 802.15.4 MAC protocol.

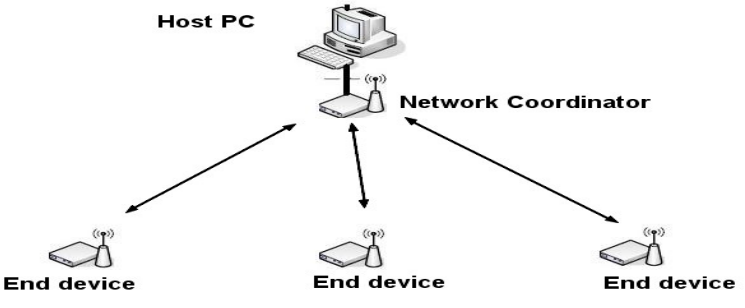

Fig. 4. Example network : Star network based in IEEE802.15.4 MAC

Following are the hardware and software used for the integration and testing:

1. A PC for development and using as a host system,
2. AllGo's wireless modules based on Freescale's MC1321x MCU, an 8 bit microcontroller of HCS08 family,
3. P&E USB multilink debugger,

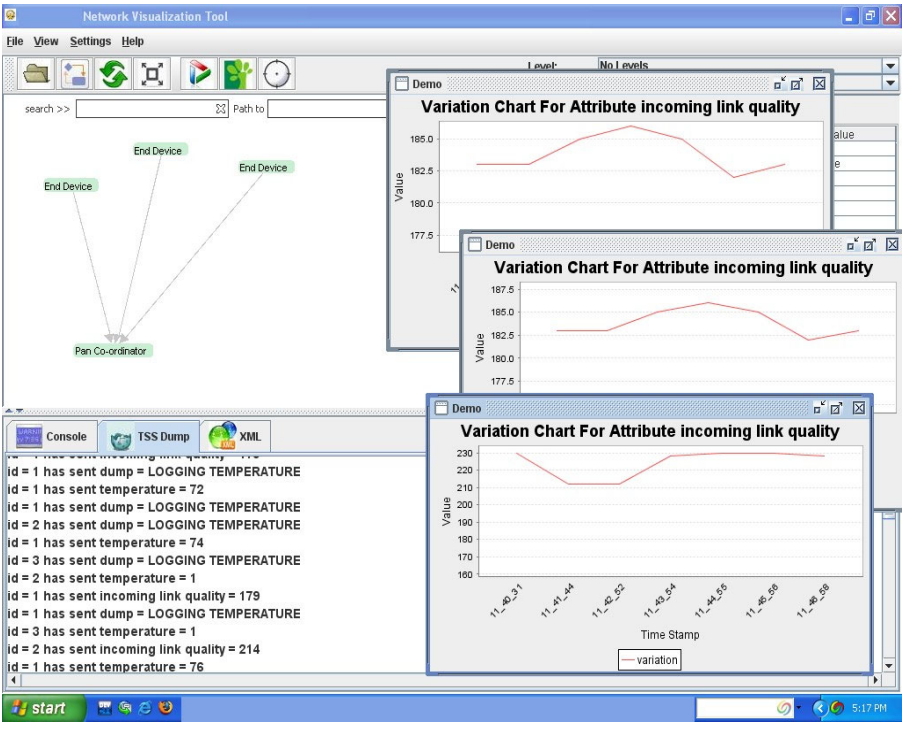

Fig. 5. Tool snapshot showing the link quality variations for each connected device

4. IEEE802.15.4 MAC code base for HCS08 generated from Freescale's Beekit,
5. JDK 1.5,
6. Prefuse toolkit for java.

A simple star network set-up as shown in Figure 4. was used for integration and testing. The network coordinator is connected to host Pc via serial port. All other devices in the network are connected to the coordinator through wireless network.

Each phase of development of this project has undergone various types of testing from the beginning of high level design till the the end of interfacing the Target Sub System part with the Host Sub System for its proper functionality. Fig.5 shows the snapshot of the tool after set up of the network.

6 Conclusions

Thus the tool can easily communicate with the application for its different purpose,. like obtaining the short address of the device, or obtaining the link quality at the data indication, or obtaining the app specific data such as temperature or other parameter to be logged etc. The tool requires the basic essential network functionality and builds its structure based on the protocol stack information. Since there is only change in the

network specific API's and Callback functions needed for the integration, this tool developed finds great importance with focus on future types of sensor networks also. The tool developed is presently tested by integrating with an IEEE802.15.4 MAC[1] based star network configuration. The following features of tool are verified systematically by the test results:

1. The basic state machine of each device is tested for proper functionality, which is the basic requirement for the tool to operate for any type of network since this is network platform independent.
2. The generic message structures developed are functioning as desired for the specific build used (802.15.4 MAC[1]). This proves the approach of designing platform specific API's and Callbacks.

Thus, this network visualization tool can become a standard framework for developing the network visualization and monitoring tool for any of the future network types also.

References

1. Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low- Rate Wireless Personal Area Networks (LR-WPANs), Institute of Electrical and Electronics Engineers, IEEE 802.15.4 (2003)
2. ISO/IEC 10039:1991, Information technology-Open systems interconnection-Local area networks- Medium Access Control (MAC) service definition (1991)
3. ISO/IEC 15802-1:1995, Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Common specifications-Part 1: Medium Access Control (MAC) service definition (1995)
4. ZigBee Specification, ZigBee Standards Organization, 053474r06 Version 1.0 (June 2005)
5. ZigBee Network layer specifications, revision 10, version 1.00, ZigBee document (2004)
6. Bulusu, N., Heidemann, J., Estrin, D.: GPS-less Low Cost Outdoor Localization For Very Small Devices. IEEE Personal Communications, Special Issue on Smart Spaces and Environments 7(5), 28–34 (2000)
7. ISO/IEC 8802-2:1998 (IEEE Std 802.2^{TM}, 1998 Edition), Information technology-Telecommunications and information exchange between systems - Local and metropolitan area networks-Specific requirements-Part 2: Logical link control (1998)
8. ISO/IEC 9646-1:1994, Information technology-Open systems interconnection-Conformance testing methodology and framework- Part 1: General concepts (1994)
9. ISO/IEC 9646-7:1995 (ITU-T Rec. X.296 (1994)), Information technology- Open systems interconnection-Conformance testing methodology and framework-Part 7: Implementation conformance statements (1995)