Anura Network Specification

Team 4

April 21st, 2013

Jonathan Bishop

Denis Coady

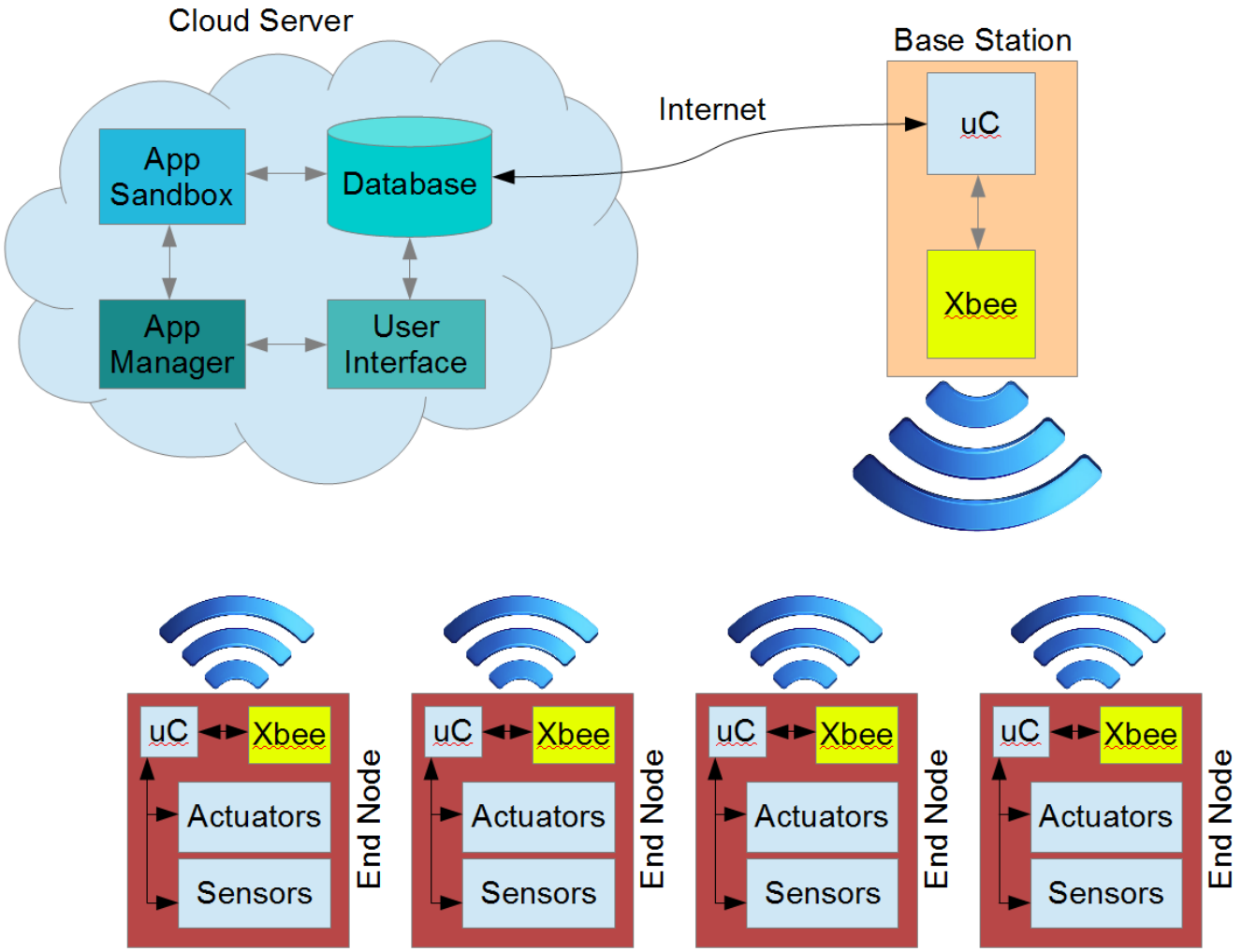
Timothy Hebert

Introduction

The Anura network is designed for use in modular home automation. The network consists of thee major device types: Cloud Server, Base Station, and End Node. By design, only one cloud server will ever exist (but as an amalgam of multiple computers depending on necessary load). This server will be the master of all internet-connected base stations; collecting and transmitting data. The base station, in turn, will be the master of all end nodes that have been paired with it. The base station will handle wireless/wired network connections and node diagnostics. The end nodes can be configured with any combination of sensors and/or actuators. The cloud server, then, can access the sensor information, and react on the environment with the actuators.

Overview

System Overview Diagram



The preceding block diagram shows how the three main devices are connected together as a network. A minimum of one cloud server, one base station, and one node are required for full operation of the system. The cloud server and any base stations are connected over the internet using TCP/IP. A given base station and any nodes are connected wirelessly using the Zigbee protocol with Xbee transcievers. All communications follow the Anura protocol outlined on page 4.

Cloud Server Details

The cloud server contains the master database, web interface, and APPs. The database will hold all data received from the end-nodes, as well as information regarding users, individual home network settings, and a list of paired nodes. It will always be internet-accessible, and will allow connections from Anura base stations for data storage. The web interface will allows users to view current data from the sensors on their network, modify the actuators on their network, and update APP settings. APPs are essentially small autonomous scripts with a given activation function related to sensor data, that produces a response by actuator and/or external interfaces. An example of an external interface would be a text-message alert.

Users can install their own APPs, written in Javascript, which run in a sandbox such that they cannot access data outside their own network, and cannot modify or delete previously-received data. They can read received data, output to actuators, and interact with any Anura-approved external interfaces through provided API.

Base Station Details

The base station acts as a bridge between the end nodes and the cloud server, passing data through as needed. It also facilitates pairing of end nodes to the home network, as well as ignoring any messages from unpaired nodes. The base station requires a wired internet connection and a wireless Xbee connection. The base station Xbee is configured as constant-on, full-power. The base station will have the functionality to ping all of its paired nodes once every 15-60 minutes to received a battery life from them, and update the cloud server with that information. If a node does not respond after two minutes, it is assumed to have run out of battery, and is marked as such in the cloud server database, which notifies the user via their set notification method.

End Node Details

End nodes will contain a selection of sensors and/or actuators. All communication to and from the device will go through an Xbee connection with the paired base station. The end-node Xbees are configured with the default cyclic-sleep option to conserve power. A given node may choose to send Write Data packets to the cloud server (through the base station) containing sensor data at any given rate. The node, however, must provide functionality such that its sensor/actuator values can be sent to the cloud server when a Read Data packet is received. They must also provide the same functionality for a percent value of battery life (ideal supply voltage vs. min operating voltage).

Protocol Details

The Anura network protocol is separated into two modes: wireless, and wired.

Wired networking is used to communicate between the base station and the cloud server. This follows standard TCP/IP, and occurs over the internet. A given packet consists of a packed JSON string. The packets are encrypted using SSL. The JSON message follows the format shown below:

```
{
    "message": {
        "type": "message_type",
        "id": "message_id"
    },
    "network": {
        "id": "network_id"
    },
    "routing": {
        "from_high": "routing_from_high",
        "from_low": "routing_from_low",
        "to_high": "routing_to_high",
        "to_low": "routing_to_low"
    },
    "data": {
        "data_name": "data_name_string",
        "type": "data_type",
        "bytes": "data_bytes"
    }
}
```

Note that this message is formatted for easy-reading. There are no carriage returns, newlines or whitespace in the actual message. All values and identifiers are strings enclosed in quotation marks. The JSON message must be ended by a null character.

The **message_type** value indicates what operation is being performed. The following message types are currently supported:

- “RD” - Read Data Message
- “RN” - Register Node Message
- “IG” - Ignore Node Message
- “AK” - Acknowledge Message
- “WD” - Write Data Message
- “PN” - Ping Message

Wireless networking is used to communicate between the base station and any paired end-nodes. This occurs over a Zigbee-standard 2.4GHz network. Xbee transceivers are used to provide a star-topology network, with the base station as the network coordinator. All network traffic is AES public/private key encrypted, with exchanges handled by the Xbees. A wireless packet consists of a 32-bit synch frame (the characters "1499"), a 32-bit JSON length in bytes, followed by the previously-mentioned packed JSON message.

Message Fields

The **message_id** value is a unique identifier for the message. This is a 32-bit unsigned integer parsed into a string. It is implemented as a rolling counter in each device, allowing specific messages to be acknowledged.

The **network_id** value is the Zigbee PAN of the network the packet originated from or is going to.

The **routing_from_high** value is the highest 32-bits of the 64-bit address of the originating device.

The **routing_from_low** value is the lowest 32-bits of the 64-bit address of the originating device.

The **routing_to_high** and **routing_to_low** values follow the same convention, but for the terminating device's address.

The **data_name_string** value is an identifier for what the data represents. E.g. "TEMPERATURE"

The **data_type** value is an identifier for the type of data being sent. E.g. "BOOLEAN"

The **data_bytes** value is the actual string-parsed value being sent.

For a RD message, this is ignored.
For a WD message, this is the data to be written to the **data_name_string** variable or field.
For an AK message, this is the **message_id** of the message being acknowledged.
For a RN message, this is the node to be registered's unique serial number.
For an IG message, this is the node to be ignored's unique serial number.
For a PN message on the wired network, this contains a timestamp.
For a PN message on the wireless network, from the base station, this contains a timestamp.
For a PN message on the wireless network, from an end node, this contains a battery %.

Pairing a Device

Base stations automatically connected to the Anura cloud server. Users can register their device with their Anura account by entering its unique identifier (from a sticker on the device) on the Anura web interface. Once registered, the cloud server sends a Register Node packet to the base station with the data_bytes field all zeros (the Zigbee coordinator's address). The packet also contains a newly-assigned PAN in the network_id field that the base station updates its Xbee with and saves internally.

End nodes must be paired individually to the base station. This is accomplished by holding the end node near the base station, and turning it on. The end-node's Xbee will automatically join the PAN with the highest signal strength coordinator, which should be the base station. Once joined, the end-node will then send a Register Node message to the base station. The base station will check its internal lookup table. If the node is already registered, it sends back a Register Node message. If the node is not already registered, it passes the Register Node message on to the cloud server. The server

then prompts the user (via the web interface) if they would like to pair the device. To proceed with a pairing, the user must verify that the displayed serial number matches the serial number on the device. Once this is done, the cloud server sends a Register Node message to the base station, the base station updates its lookup table, and resends the Register Node message to the end node to complete the process. A user may choose to ignore a node that is attempting to pair to their network, which will prompt the cloud server to send an Ignore Node message to the base station. The base station will update a second lookup table for nodes to be ignored, and then resend the Ignore Node message to the node attempting to pair. Upon receipt of an Ignore Node message, an end-node effectively becomes disabled until it is power-cycled.

The user can choose to remove devices from the ignore list and paired list at any time via the web interface.

Document History

Reviser	Date	Description
J. Bishop	04/21/13	Initial draft