Wireless Mesh Networking - An Open Approach

Brar, Harman Paul S., Foster, Kyle B., Greentree, Abigail R.,
Nelson, Malik V., Ramirez, Ethan C., and Weinstein, Alexander S.
Booz Allen Hamilton - McLean, VA

There is no doubt that the software-defined radio or "SDR" will become the wireless communications platform of the future. While the concept has been around since the seventies, the hardware technology is just now becoming capable of the task. A natural application for the "SDR' with its versatility, powerful digital signal processing capabilities and small form factor is within an adaptive wireless mesh network topology. In this summary, we will be describing our approach and proof-of-concept wireless mesh protocol emulated with the open-source microcontroller, Arduino and Nordic Semiconductor's NRF24L01+, a well documented wireless transceiver.

INTRODUCTION

Military theaters in general consist not of obstruction free, wide open planes but of difficult to navigate labyrinths. Recently, these environments have consisted of mountainous terrains and urban layouts; to add to the challenge, uncontrollable variables such as weather also add a layer of complexity. It is the task of the engineer to design systems able to react to such challenges and perform to specification at all times. Of course this is not an ideal world and therefore such an ideal system may never exist, but in this description, a proposal is made for a system to prove the concept of a clandestine wireless mesh network operating autonomously in the field of battle. It is designed to satisfy the "Contingency" and "Emergency" operations in the military's "P.A.C.E" communication protocol, but given the constant advancement of software-defined radio technology, a system such as this may also be adapted to satisfy the "Primary" and "Alternate" functions as well. Below, we will discuss the basic functionality of the developed protocol and describe how it was implemented.

THE CIRCUIT

The system is chiefly digital and low speed, therefore the considerations needed to construct a healthy circuit are quite minimal. The nodes are to be supplied by a standard nine volt alkaline battery, which feeds an SOT-223 5VDC linear regulator on -board the Arduino. This in turn feeds a TO-220 3.3VDC linear regulator who supplies the power for the NRF24L01+. While the 3.3VDC pin on the Arduino can supply up to 150mA, users have found reliability issues with the NRF24l01+, solved by using a separate 3.3VDC regulator with reservoir capacitors before and after; 47μF was chosen simply due to availability.

The NRF24L01+, TO-220 3.3VDC regulator, and capacitors were mounted upon a proto-shield designed to sit on top of the Arduino I/O headers. Two 3mm red

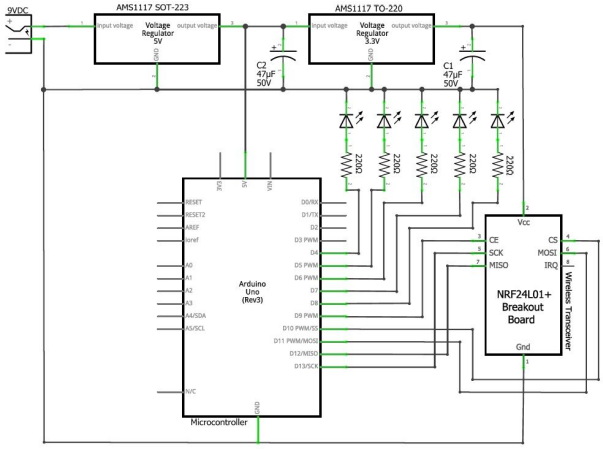

FIG. 1. Node Schematic

LEDs were pre-mounted on the proto-shield and three 5mm LEDs were added to give visual feedback to what is currently happening with the node while it is running headless: one flash of the 3mm LEDs every time the node sends out a broadcast, rapid flashing of all LEDs when data is being routed through the node, and periodic pulsing of the 3mm LEDs when the node is off the network. In terms of pin choices, the Arduino uses digital I/O pins zero and one for serial communication with the computer to upload the sketch and interact with the Serial Monitor, and digital I/O pins two and three for hardware interrupts. If possible, do not connect any outputs to DIO pins zero and one. Each connected component adds to the line capacitance which resists the change in line voltage during data transmission, thus causing sketch uploads to fail. DIO pins two and three were reserved for potential use with a pushbutton to generate an interrupt and run an alternate routine.

THE PROTOCOL

The protocol defines a shared mesh network topology for wireless data transmission. It features the ba-

FIG. 2. Actual Node

sic functionalities of a mesh network including data relay through multiple nodes, automatic topology maintenance, prevention of routing loops, and rudimentary path optimization all achieved on hardware supporting half-duplex transmission. To build this foundation, a header consisting of five of the thirty-two byte payload is constructed to transfer pertinent information regarding the transmission occurring between nodes.

MSB				LSB
to node address	from node address	final node address	message lifetime	message type
2 HEX	2 HEX	2 HEX	uint_8	1 char
1 byte	1 byte	1 byte	1 byte	1 byte

FIG. 3. Message Header

Header Partitioning:

1. The “to node” address is the address of the node you want to receive the transmission; this could be a final node or a routing node. Note: “to node” == “final node” iff the recipient is directly connected to the sender.

2. The “from node” address is the address of the original sender. This is the node that generated the message.

3. The “final node” address is the address of the destination node. This is the only node who will be able to open and read the message.

4. Message lifetime implements a counter that decrements every time it is forwarded or “hops” to prevent routing loops and network flooding. The value is an unsigned integer so the maximum count is two hundred and fifty-five hops, but should be set to one less than the number of nodes in the network.

5. Message type carries a char ‘M’ for “message” type transmission or a ‘B’ for a “broadcast” type transmission.

The “from node” address, “final node” address, and message type are set by the original sender will not be changed during the lifetime of the message. Thus, the “to node” address and message lifetime will be updated by the routing nodes that the message passes through. When data is routed, the transmitting node will update the “to node” address to be the next node that should receive the payload and will decrement message lifetime.

The remaining twenty-seven bytes (after the LSB) are left to transport ASCII coded characters ie. the message. If the message to be sent is greater than twenty-seven characters, the system truncates the message and drops the remainder. Extensions can be made to split the message into appropriately sized chunks and send consecutive messages.

The prevention of routing loops relies on three principles:

1. “Message lifetime” routine: A message may propagate through the network for a total amount of relays described by this value. When the value reaches zero, the last recipient discards the message, ceasing the forwarding routine and preventing the network from becoming flooded. This has the greatest effect when the recipient is not currently on the network which would cause the message to endlessly search for its destination.

2. “Direct routing priority” (DRP): Whenever data is being transmitted, either by the original sender or a routing node, the protocol executes an algorithm to decide which node to send the data to. The first step of this algorithm is DRP, which checks if that node is connected to the final node. If it is connected, then the to-node address is set to be the address of the final node. This means that messages are sent directly whenever possible. This ensures that messages will not be looped between nodes who are each other’s primary connections.

3. Method of “no return”: Upon receipt of a message, the system determines who the message is for and decides to keep it or pass it on. In the event of the message being relayed, the protocol would automatically send via the top node on the stack, but if this node is where the message originated, the next node in the stack will be selected as the primary route to prevent the message being continually passed back and forth between two neighbors who are each others node(1).

The “broadcast” routine maintains the chronological stack and therefore maintains the one-hop neighbor topology for each node. This method leverages the self similarity characteristic of a multi-cast network topology with the added feature of methodically selecting the neighbor who provides the most reliable path to the destination. This therefore saves on power consumption of

the transmitting node and minimizes the per-node network bandwidth consumption. Together, these allow for an increased total data throughput on a shared frequency band.

Due to the potential latency involved with the natural redundancy of a mesh network, path optimization and route reliability need to be addressed. To do so the protocol leverages the DRP and its chronological stack. The protocol first checks the stack to see if it has recently been connected to the destination node, if so, the message is sent directly. This step will be performed at each retransmission point which minimizes hops and ensures that the data takes the most direct route. If the data cannot be sent directly, the message is forwarded on to node(1) on the stack.

The properties of this protocol allow it to keep itself up to date with the local topology and adjust for changes at a rate equal to the frequency of the "broadcast" method. This protocol lays a foundation satisfying the requisite behavior of a rudimentary half-duplex mesh network and given more processing power or a chip supporting multiple parallel processes, the existing features can easily be richened to provide an even more effective platform.

CLOSING REMARKS

Hardware reliability: Remained consistent aside from a few intermittent issues where an NRF24L01+ would cease to communicate with the host Arduino, only to resume operation at a later time with no deducible reason. The reliability heightened however once the nodes were soldered together as opposed to connections being made by male-to-female jumper wires.

Node Performance: The NRF24L01+ has adjustable output power and data rate which together directly influence the maximum transmission range. In the best case scenario, the output power was set at 0dBm and data rate set to 250kbps, yielding a maximum confirmed range of 55 meters line-of-sight. The signal at these settings was also quite consistent through multiple wall partitions in an office environment, though no quantitative measurements were made for this metric.

Note: Attached at the end is an appendix containing the state diagram and all of the flow graphs describing the logical flow of each function. Labels and terminology was chosen to be readable but also maintain loyalty to our protocol.

We would like to thank Booz Allen Hamilton for its support of the Summer Games internship and our project leaders, Varun Sarin and Henok "Henny" Girmai, for their guidance throughout the project.