SSC17-VII-10

Mesh Network Architecture for Enabling Inter-Spacecraft Communication

Christopher Becker, Garrick Merrill
NASA Marshall Space Flight Center
Mail Stop EV42, Huntsville, AL 35812
chris.becker@nasa.gov

ABSTRACT

To enable communication between spacecraft operating in a formation or small constellation, a mesh network architecture was developed and tested using a time division multiple access (TDMA) communication scheme. The network is designed to allow for the exchange of telemetry and other data between spacecraft to enable collaboration between small spacecraft. The system uses a peer-to-peer topology with no central router, removing the possibility of a single point of failure. The mesh network is dynamically configurable to allow for addition and subtraction of new spacecraft into the communication network. Flight testing was performed using an unmanned aerial system (UAS) formation acting as a spacecraft analogue and providing a stressing environment to prove out mesh network performance. The mesh network was primarily devised to provide low latency, high frequency communication but is flexible and can also be configured to provide higher bandwidth for applications desiring high data throughput. The network includes a relay functionality that extends the maximum range between spacecraft in the network by relaying data from node to node. The mesh network control is implemented completely in software making it hardware agnostic, thereby allowing it to function with a wide variety of existing radios and computing platforms.

INTRODUCTION

As the use of cubesats and other small satellites continues to grow, communicating with the larger numbers of on-orbit assets will start to stress ground communications capabilities. In addition to single satellite missions, multiple organizations have begun to develop and deploy constellations of satellites and more are planned in the near future. To help relieve the demands being placed on ground stations and to enable communication between satellites, a TDMA-based mesh network communication system was developed. This system uses a peer-to-peer architecture and does not require a central master node or router. This eliminates the possibility of single-point failures due to the loss of the network master.

The designed mesh network allows a small formation of satellites to collaborate and exchange data to enable their mission and reduce ground communication requirements. By exchanging data directly with other satellites in the formation, the formation can function more autonomously with less ground intervention required. The communication system was designed to be reconfigurable for different applications. Some of the driving design goals were to have low latency, to allow for addition and removal of communication nodes in the network without interruption, to relay data across the network, and to make the communication architecture hardware agnostic, not requiring it to be dependent on a specific hardware implementation.

NETWORK DESCRIPTION

The mesh network functions by assigning time blocks to individual network nodes. A node is any entity communicating using the mesh network protocol. The time allocations are determined using a time division multiple access-based architecture. This architecture is illustrated in Figure 1. Time is sliced in segments called Frames. A Frame consists of the Cycle and the Sleep periods. Primary communication across the network is performed during the Cycle. The Cycle is broken down into Slots, where a Slot is the portion of time provided to each node to perform its outgoing communication on the network. The Frame, Cycle, and Slot lengths are all configurable parameters.

During a Slot, only one node is transmitting, and all other nodes are listening. To ensure data integrity and accommodate some variation of clock times across the network, delay periods are built into the communication protocol. As shown in the figure, this pattern of delays is designed to ensure that receiving nodes are listening for the entirety of the transmitting node's transmission. Once a node is done transmitting, it will change over into receive mode and prepare to listen to other transmitting nodes. The lengths of the sub-periods within the slot are configurable, so as to make the network architecture flexible for specific applications.

Once all slots are completed, the Cycle ends and a Sleep period begins for the remaining time in the Frame. During the Sleep, all nodes are nominally

quiescent, allowing for power-savings when communication is not necessary. The Sleep period could also be used for aperiodic communications or network administration such as reconfiguring the parameters of the mesh network protocol.

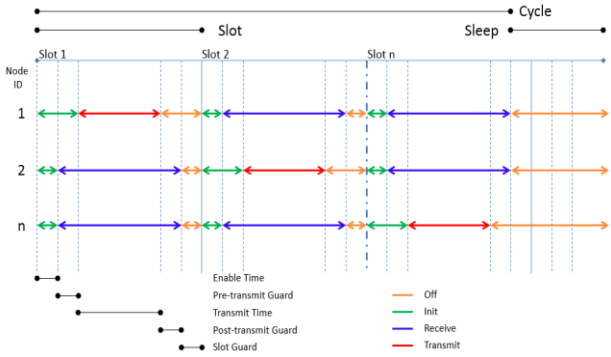

Figure 1. Mesh Network TDMA Frame

To function without a master node, the nodes in the network require a common time source to maintain the integrity of the TDMA architecture. The system is not dependent on a specific method of synchronizing time, but the reference design was developed and tested assuming the individual node clocks are synced using time received from the Global Positioning System (GPS). Since GPS is widely used already by spacecraft for orbital position and other data, it is a convenient, readily available, and reliable time source.

Network Topology and Data Relay

The network topology employed is a simple point-to-point design, as illustrated in Figure 2. When nodes broadcast, all other nodes in range receive the data. Any nodes not in range of a broadcasting node will not receive its data directly during the initial transmission.

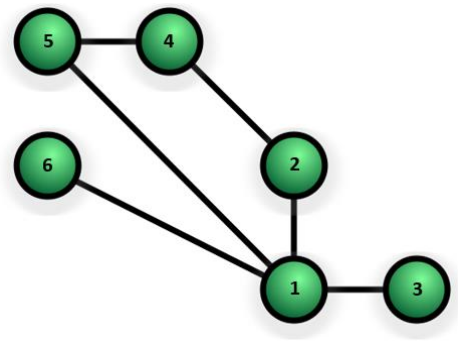

Figure 2. Mesh Network Topology

However, using the data relay capability in the network, nodes will transmit not only their own data, but also data received from other nodes. The relay functionality is performed in a single repeat manner, meaning that nodes will only rebroadcast unique data once. For example, for the network shown in Figure 2, Node 2 would receive data from Nodes 1 and 4 directly. When Node 2 enters its transmission period, it would pass its own data and any data received from Node 1 back out to be received by Nodes 1 and 4. When Node 1 receives this transmission from Node 2, it will recognize the portion of the message that it originally transmitted. The next time that Node 1 transmits, it will not retransmit that portion of the data again. This prevents data that was previously sent from being relayed back and forth across the network endlessly.

Because there is no master node, the network will continue to function regardless of what specific nodes are currently present in the network. Any node present will transmit during its allotted Slot and receive data from other nodes during their Slots. If a previously present node drops out of the network, the other nodes will notice the data dropout during the lost node's Slot, but the network will remain intact for usage by the remaining nodes. Since the network topology is point-to-point, any node that couldn't communicate directly with other network nodes without going through the lost node will become isolated. For example, if Node 1 dropped out of the network, Nodes 3 and 6 would also lose communication with the network in Figure 2.

DEVELOPMENT

The mesh network architecture was developed in stages over the course of several years. Initial development began in Fiscal Year 2014 (October 2013-September 2014) with initial basic functionality depending on commercial of the shelf (COTS) systems. In the following years, new features were added to expand the capabilities of the network and the test hardware implementation was altered to enable these features.

This iterative development approach involved both software development of the mesh network architecture as well as hardware implementations used as testbeds for the network. The software and hardware development is described in the following two sections.

Software

Because of the full Linux development environment afforded by the BeagleBone Black used in the hardware implementations described in the following section, primary software development was performed in Python. This afforded the developers with a flexible software development environment to quickly create, adapt, and test new features. The software was developed with modularity in mind, so that it could be modified for use with a wide variety of radios and hardware implementations. The software was designed

using object-oriented processes allowing hardware specific code to inherit from the generic codebase.

Network configuration for a specific application is performed using a JSON-based configuration file that contains configurable parameter values used during execution of the mesh network code. Configuration parameters can also include flight vehicle specific parameters such as radio interfaces and settings. By placing configuration settings in an easily modified human-readable file, this allows for quick reconfiguration of the software without having to modify the source code. This reconfigurability allows the network performance and behavior to be catered for specific applications, such as modifying the network to prioritize data throughput over low latency for science operations that generate a large amount of data.

For the current generation of hardware which uses an FPGA (field-programmable gate array), the Python mesh network logic was ported into VHDL (VHSIC Hardware Description Language). This included the mesh network control itself as well as the time synchronization functions, specifically the interface to the GPS. Initial development of a C++ implementation that would be more suitable for actual flight code usage has also been created.

Hardware

The hardware implementations described below were developed to function separately from the main vehicle flight computer with their own independent hardware and software for modularity purposes, but the mesh network software could also be deployed to run directly on the host vehicle's flight computer. The reference hardware systems described in the following subsections used standalone radios, but existing radios on the host platform could also be used assuming the necessary bandwidth was available and the radios were suitable for providing the required coordination and timing.

First Generation

Preliminary mesh network development began by exploring available options for the communications link. Initial options were explored based on their suitability for immediate testing and not necessarily their applicability for the final design. Network layer options explored included WiFi and existing COTS personal area network technologies such as ZigBee. Because of its simplicity, wide availability, and inter-operability with other existing test equipment, initial design studies converged on the use of XBee radios.

The first generation node hardware consisted of a BeagleBone Black (BBB) single-board computer, two XBee Pro 2.4GHz radios, and a custom BBB interface board called a “cape” (Figure 3). The XBee radios are attached to headers on the cape which in turn mate to the headers on the BBB. The first generation network design used two independent mesh networks operating on different frequencies to provide redundancy.

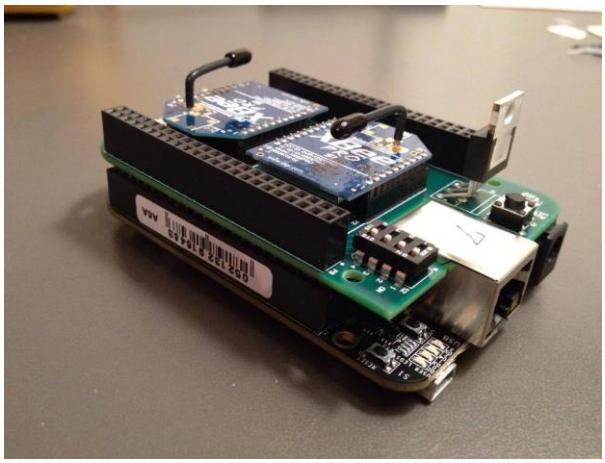

Figure 3. First Generation Node Stack

This first generation network was based on a proprietary XBee networking protocol.[1] This protocol was responsible for coordinating and controlling communication across the network. The TDMA mesh network scheme that is the primary topic of this paper had not yet been developed. By leveraging existing technology, we were able to quickly create an initial functioning system and concentrate on developing our testing architecture. This initial generation also did not yet employ relay functionality, requiring all nodes to be in direct communication with all other nodes to ensure network integrity.

The BBB in the hardware node stack interfaced with the test vehicle’s flight computer via an RS-232 serial UART (Universal Asynchronous Receiver/Transmitter). Python scripts were developed and run on the BBB to interface with both the flight computer and the radios and to compile and process data for passing over the communication network. All data was transmitted over both redundant networks with duplicate data being parsed and discarded by the receiving nodes.

Second Generation

The second generation system was designed to make the communication system hardware independent, so that it would not be dependent on a particular model or brand of radio to function. To enable this, a custom TDMA scheme was developed to control the sequencing of communication on the network (Figure 1). This contrasts with the first generation system

which did not have any software-based communication control scheme but instead relied on the XBee radios to provide this function. By moving this function into software, the system is not only more hardware-independent, but the TDMA scheme also helps cut down on power requirements by allowing the radio receiver and transmitter to be powered off when not in use.

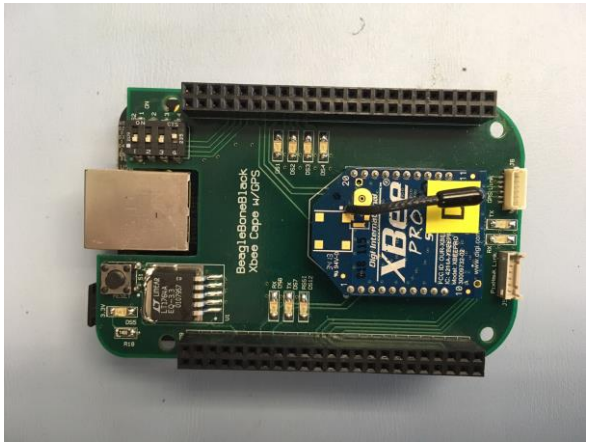

Figure 4. Second Generation Xbee Node Stack

Figure 5. Second Generation AstroDev Li-1 Node Stack

To further reduce power requirements as well as mass, only one network is employed therefore only requiring one radio. To showcase the capabilities of the system and to demonstrate deployment on a wide range of hardware, a relatively simple radio with minimum complexity was desired. The radio hardware requirements were also simplified by moving the collision and other communication control logic into the mesh network communication system software.

The second generation system also added data relaying to allow nodes to communicate and pass data and commands between all vehicles without requiring direct communication between all nodes. Relaying allows commands and data to propagate along the mesh network to any desired destination node, allowing a node that has no direct communication path to a particular node to still receive that node's data.

Implementation of the TDMA scheme also required precise timing, so a method had to be provided to synchronize the clocks of all nodes in the system. Because of its existing widespread use as a vehicle navigation source by many vehicle types, GPS was chosen as the time synchronization source. The time broadcast by the GPS constellation and a pulse per second (PPS) signal from a GPS receiver are used to provide time synchronization within 1 millisecond or better across the network nodes. However the communication system is not dependent on this particular time source, so any other external time synchronization method implemented by the host platform would be sufficient provided it meets the time accuracy requirements.

The new TDMA scheme was tested on two different hardware implementations. The first was a modification of the first generation XBee stack but removing the second radio and adding a GPS interface (Figure 4). To show that the mesh network would function using a flight-ready radio, a hardware implementation using the AstroDev Li-1 UHF radio was also tested (Figure 5).

Current Generation

During development of the current generation of the network, the primary goal was to further improve network timing to ensure the most efficient usage of available communication bandwidth. To facilitate improved timing accuracy, an FPGA was added into the hardware implementation. By moving the network timing and control logic onto the embedded FPGA, time critical events, such as the start of each individual time segment of the TDMA Frame architecture, could rely on the more precise, repeatable execution afforded by the FPGA versus running it on a general purpose microprocessor such as the vehicle's flight computer. This precision then afforded to the option to reduce the lengths of the delay periods introduced into the mesh network architecture to account for less specific timing.

The FPGA used was a Microsemi ProASIC3, chosen because it provides a path towards space-quality hardware. The BeagleBone Black platform was retained as in the previous generations just with the new interface board (Figure 6). The TDMA network logic as well as the interface with the GPS was moved into VHDL running on the FPGA. The data processing and other interface functions, such as communicating with

the flight computer, were retained as Python scripts running on the BBB.

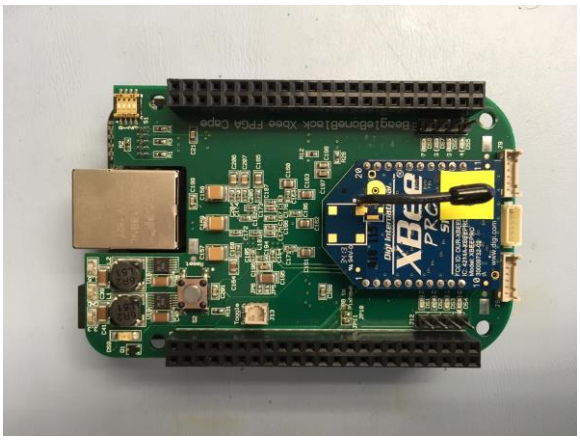

Figure 6. Third Generation XBee Node Stack

TESTING

Testing was performed in three phases. Testing began with initial mesh network software development and debugging in a lab environment. Figure 7 shows testing of 8 second generation XBee nodes. The node stacks are shown with their attached GPS receivers for time synchronization purposes and interfaces to simulated flight computers. In lab testing, the nodes were interfaced to simulated vehicle flight computers, via serial link, emulated by a desktop PC. This allowed quick modeling of mesh network behavior in different test scenarios.

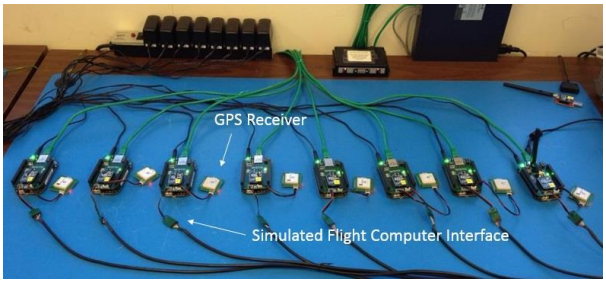

Figure 7. Lab Test Setup

Upon completion of initial lab testing, network testing then moved towards testing formation applications involving two primary stages: a flight test program to demonstrate the network's capabilities in a realistic, dynamic scenario and a simulation environment to test spacecraft implementation specifics. To provide a dynamic and stressing test environment for the mesh network, small unmanned aerial systems (UAS) were used as a flight test platform. Small consumer-grade quadcopters were chosen because they were readily available to the development engineers. Each quadcopter carried the BBB node stack as a payload (Figure 8). The quadcopters were operated in formations with up to six vehicles in flight simultaneously. The node stack interfaced with the flight computer of the quadcopter via serial link to provide vehicle state data for transmission over the mesh network and to transmit vehicle control commands to the flight computer.

The UAS test regime was comparatively more stressing than a typical on-orbit spacecraft application and required a network with low latency. With the UAS moving quickly and in close proximity (within a couple meters of one another), a UAS formation provided an ideal test case for stressing the mesh network and ensuring it was stable and robust.

Figure 8. Quadcopter Test Vehicle with Mesh Network Payload

Flight Testing

Flight tests began with a single vehicle node and a ground station that was also a node in the mesh network. Single vehicle tests were used to show that the mesh network could operate successfully in an environment outside of the lab and also provided for testing of the communication links to ensure necessary range for the flight testing.

Testing then proceeded by gradually adding more and more vehicles to the flight tests to increase the stress on the mesh network. In parallel to the actual mesh network software development, logic had to be developed to control the UAS formation movement itself. A formation control scheme was developed that allowed the vehicles to safely and collaboratively move in close proximity to one another while ensuring that collisions would be avoided.

While the specifics of this scheme are not relevant to the mesh network itself, it was a necessary element to enable a realistic test environment and to flesh out deficiencies in the mesh network implementation. The

formation scheme developed allowed for safe flight of a large number of vehicles (Figure 9), limited only in the scope of these tests by the number of vehicles available. Ultimately flights of up to six quadcopters were achieved, and including the ground station, this demonstrated the functionality of the designed mesh network with seven total nodes. Video of a five-vehicle flight can be found here: https://www.youtube.com/watch?v=oH9C43To3Dk .

Figure 9. Five Vehicle Flight Test

For flight tests, the mesh network was operated with a TDMA Frame of one second. The Frame was split into equal Slot sizes for all seven nodes (six flight vehicles plus the ground station). This Frame size was chosen to reduce communication latency between the vehicles. The short Frame length, combined with the relatively low bandwidth of the radios under test (115200 baud max), resulted in a small individual data throughput for each node. However since all of the TDMA network parameters are configurable, the network is easily adaptable based on the requirements of the specific application. If total data throughput is a higher priority, the data to be transmitted could either be broken into smaller pieces for transmission, or the Frame length could simply be lengthened allowing more contiguous data transmission. Likewise a higher bandwidth radio could be chosen.

Spacecraft Simulation

To test the usage of the mesh network in a scenario relevant to spacecraft applications, a spacecraft test simulation was developed. The spacecraft simulation was performed in the lab without any modeling of communication latencies due to distance between network nodes. The primary purpose of the satellite simulation was to show that the network was capable of transmitting the necessary types of data that would be required by a real spacecraft formation. The formation modeled employed an eccentricity/inclination vector separation technique to control relative motion between a chief spacecraft and several deputy satellites.[2] By exchanging information across the mesh network, the individual nodes were made aware of spacecraft position deviations from the ideal planned trajectory, due to environmental disturbance forces, and could correct for them.

While the UAS test flights showed the ability of the network to react in a situation requiring low latency, the satellite simulations tested the longer term stability of the network. Real-time simulations of up to 3 days were run to show that the communication links were maintained and the network remained stable using the TDMA architecture. The simulated spacecraft were able to exchange data and maintain the desired formation spacing.

FUTURE DEVELOPMENT

Future development goals include expanding the capabilities of the network to self-adapt to changing network conditions and loss/addition of new network nodes. Currently the network is highly configurable but relies on a priori knowledge of the expected size of the network and data throughput needs to create a configuration file used by all network nodes. This file which is loaded by each node when it joins the network contains all the configuration parameters of the network such as Frame and Slot lengths. By allowing the network nodes to change the specific TDMA timing settings dynamically, the network can make more efficient usage of the available bandwidth, in the case for example of the loss of a network node. The lost network node's Slot time could be reallocated allowing longer Slots for each of the remaining nodes. While this can be achieved currently, it requires the current network to be dissolved and reinitiated.

Other future efforts will include configuring the hardware implementation for flight, so that it could be available when a future cubesat or other small satellite opportunity becomes available. This effort will include choosing hardware components that can survive the orbit environment as well as any customization required to interface with a specific mission's components such as flight computer or radios. Additionally, if the FPGA design is chosen for spaceflight testing, further ancillary logic currently being handled on the BeagleBone Black, such as interfacing with the flight computer and radios, will need to be moved into VHDL on the FPGA.

CONCLUSION

A mesh network architecture and implementation was developed and demonstrated that allows a formation or small constellation of spacecraft to communicate amongst themselves to achieve mission goals and reduce reliance on ground communication assets. The first generation system demonstrated the basic network concept was viable and allowed exchange of relevant data between network nodes. Generations two and three evolved the network design by moving it entirely

into software and/or firmware, allowing it to be independent of any specific hardware implementation. The network was tested, and it demonstrated low latency, reliable communication enabling flight tests of a small UAS formation.

The mesh network was designed to be flexible and easily reconfigurable to meet the communication needs of a wide variety of spacecraft formations and applications. The primary unique aspect of this mesh network is its lack of a master node or router. Instead the network nodes all function as peers, allowing the system to function even after the failure or loss of any node.

The mesh network software and logic described in this paper (Python, VHDL, and C++) are available from the NASA software catalog, https://software.nasa.gov/software/MFS-33391-1, and NASA GitHub, https://github.com/nasa/meshNetwork.

References

1. "Digi XBee DigiMesh 2.4 Wireless Mesh Networking RF Module," https://www.digi.com/products/xbee-rf-solutions/embedded-rf-modules-modems/xbee-digimesh-2-4, May 2017.
2. D'Amico, S. and O. Montenbruck, "Proximity Operations of Formation-Flying Spacecraft Using an Eccentricity/Inclination Vector Separation," Journal of Guidance, Control, & Dynamics, vol 29, No. 3, May-June 2006.