OPTIMIZED COMMUNICATIONS PROTOCOL FOR LOW EARTH ORBIT CUBESAT

A Thesis
Presented to
The Academic Faculty

by

Baijun Desai

In Partial Fulfillment
of the Requirements for the Degree
B.S in Computer Science with the Research Option in the
School of Computer Science

Georgia Institute of Technology
May 2018

OPTIMIZED COMMUNICATIONS PROTOCOL FOR LOW EARTH ORBIT CUBESAT

Approved by:

Dr. Marcus Holzinger, Advisor
School of Aerospace
Georgia Institute of Technology

Dr. Mostafa Ammar
School of Computer Science
Georgia Institute of Technology

Date Approved: [Date Approved by Committee]

ACKNOWLEDGEMENTS

I wish to thank Dr. Marcus Holzinger, professor at the School of Aerospace, for allowing me to conduct my research on the ongoing RECONSO CubeSat project. I would also like to thank Dr. Mostafa Ammar, professor at the School of Computer Science for guiding me in writing a thesis on a communications protocol.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

LEO	Low Earth Orbit
UHF	Ultra High Frequency
TNC	Terminal Node Controller
CFDP	CCSDS File Delivery Protocol
SCPS	Space Communications Protocol Specifications
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
RSFDP	Repeated Sending File Delivery Protocol
SRMA	Scheduled-Retransmission Multiaccess
CSP	Cubesat Space Protocol
SDR	Software Defined Radio

SUMMARY

The purpose of this document is to describe and analyze the implementation of a communications protocol to transfer data between a CubeSat and ground station. The mission of the CubeSat is to track debris in low earth orbit (LEO). Data transfers to and from the satellite include images, telemetry, commands, and processed data. The communications channel operates on a half-duplex connection with a single linearly polarized half-wave UHF dipole attached to the CubeSat, and a circularly polarized Yagi-Uda antenna on the ground station. Data will pass through a Software Defined Radio (SDR) and on the ground station over packet radio. The challenges faced in designing the protocol include high packet loss, short and infrequent access times, high delay times, variable signal strength, and limited power. The designed protocol will be implemented and tested on the CubeSat. It is evaluated against other existing communications protocols for performance.

CHAPTER 1

INTRODUCTION

A networking protocol is a standard used to facilitate communications between endpoints over a network of arbitrary size. The protocols are divided into five layers by the responsibility of the protocol and are listed in the following few sentences from higher level layers to lower level layers. The application layer gives users the ability to transfer information over the network. The transport layer is involved in formatting data, error checking, and delivery. The network layer is responsible for routing the data to the correct location in the network. The data-link layer interfaces with the physical network to send out data. The physical layer involves the hardware used to transfer data. These layers have the ability to structure data into packets and send them to other layers. Transmitting packets from device to device may lead to errors in transfer such as packet loss, bit error, duplicate packets, and out of order packets, which lead to adverse effects in communication time.

Many optimized networking protocols exist for modern day networks. There are not as many protocols optimized to communicate with devices from Earth to LEO In this type of environment, there is a significant amount of propagation delay, possibility for bit errors, and there may also be only small windows for transmitting data because of properties of the orbit [1]. This brings about the need for protocols which can handle all of these factors while maintaining an acceptable data transfer rate, which is what this paper intends to investigate.

There are a few existing protocols in existence for space communications such as CCSDS File Delivery Protocol (CFDP) and a few extensions to existing protocols such as Space Communications Protocol Specifications (SCPS) created by the Consultative Committee for Space Data Systems. Reliable CFDP built on Transmission Control

Protocol (TCP) has been shown to degrade performance more than standalone TCP, which is already not suited for space communication. When it is built on User Datagram Protocol (UDP) however, there is a less significant packet loss [2]. SCPS provides extensions to existing protocols, which optimize packet transfer over environments such as LEO [2]. Both of these however, rely on the use of TCP or UDP in the transport layer for communication, which may prevent optimal performance.

This paper will investigate methods to communicate data and optimize goodput in high loss, high delay environments such as space and attempt to implement protocols which do not necessarily rely on existing transport protocols. This method could lead to more optimal performance because the protocols may be built from the port level with a LEO environment in mind. Tests will be run on these protocols by introducing propagation delay, packet loss, and bit errors to simulate a LEO environment.

CHAPTER 2

LITERATURE REVIEW

Satellite communication between Earth and space is difficult to achieve efficiently through traditional networking methods with a low data bandwidth, small memory, and a small window of allowed transmission time. Transport protocols such as reliable Transmission Control Protocol (TCP) and application layer protocols which are built on top of it such as FTP must be modified or discarded in this environment due to large delay times, high bit error rates, and intermittent connectivity [1]. This is due to the fact that every error in packet contents or order will cause a retransmission, which is costly due to both delay and connectivity. One proposed alternative by Wang, Li, Chen, and Wu is Repeated Sending File Delivery Protocol (RSFDP) which is an adaptive protocol that adjusts retransmission parameters based on the bit error rate. It uses statistics on errors and latency to make changes in future transmissions so that retransmission is less frequent. While this may approach an optimal retransmission rate, it is not guaranteed, especially when there are large variances in what this rate should be. It also does not solve the problem of bit errors being introduced in the transmission.

Another factor that must be taken into consideration for many satellites, especially those communicating in half duplex, is packet collision. This can occur when one end of the communication link is attempting to transmit at the same time it is supposed to receive a packet. Scheduled-Retransmission Multiaccess (SRMA) by Tak-Shing and Wong is designed to reduce this effect [3]. It divides a packet into frames, with status vectors which are used for acknowledgement of packet delivery. If the satellite were to communicate to a station and either the satellite or the station is unable to verify

acknowledgement of packet delivery, a retransmission occurs. This type of approach solves the problem of reliability of packet transfer, however it sacrifices valuable time that the satellite needs in order to communicate with the station because of the amount of retransmission.

Hadjiyiannis et al propose an interesting solution to solving the problem of low power and low bandwidth by taking into account the type of data being communicated [4]. Since they were experimenting with multimedia data, packets had shapes associated with them. This allowed for error correction on the application layer that decreases the amount of time that transmissions need to occur due to bit errors while also reducing power consumption. There were a few drawbacks in this research however. The researchers were not able to take any measurements in the latency of transmission. They were also running error correction on a very specialized case. With more random data than shapes, it would not be so easy to extrapolate a fix for bit errors.

Pujari et al state that a low network bandwidth problem can be mitigated by selectively loading only the things the user needs [5]. This paper was written in the context of the internet, which does not currently selectively load data usually, but it can also be applied to loading content such as images and telemetry from satellites. Their approach is to assign priorities to which data needs to be sent first. This will ensure that the user receives the most essential data through each data transfer. This can also be done on a satellite where requests on data transmission will send only the most important telemetry or data as indicated by the user. This makes it likely that important data will not be lost in transmission, however it does not effectively reduce the retransmission rate issue which this research aims to mitigate.

A protocol designed specifically for space communication, CFDP, was analyzed by Wang et al [2]. They demonstrate that CFDP produced worse results than TCP in terms of throughput. This implies the need for a better protocol since TCP is already determined to be an undesirable protocol due to the round-trip delays and retransmission [2]. The current research aims to come up with a protocol that should outperform TCP in a high stressed communication environment in space.

The protocols seen in this review all have tradeoffs in either data rate, latency, bit errors, and power for communication. The current research aims to develop a protocol for communicating with a satellite in LEO that will attempt to optimize these tradeoffs. By using parts these protocols as well as other existing ones, this study aims to create a protocol that will allow for efficient communication to a satellite under a highly stressed communication environment.

CHAPTER 3

MATERIALS AND METHODS

3.1 Ground Station Setup

The hardware used for testing of transport protocols was the same setup that is to be used during mission. A magnetic mount 430 MHz UHF monopole with a maximum gain of 4.15 dBi was connected to a Ettus USRP B200 SDR . The SDR was connected to the USB port on a machine running Ubuntu 16.04 to emulate the ground station while the satellite is on the ground.

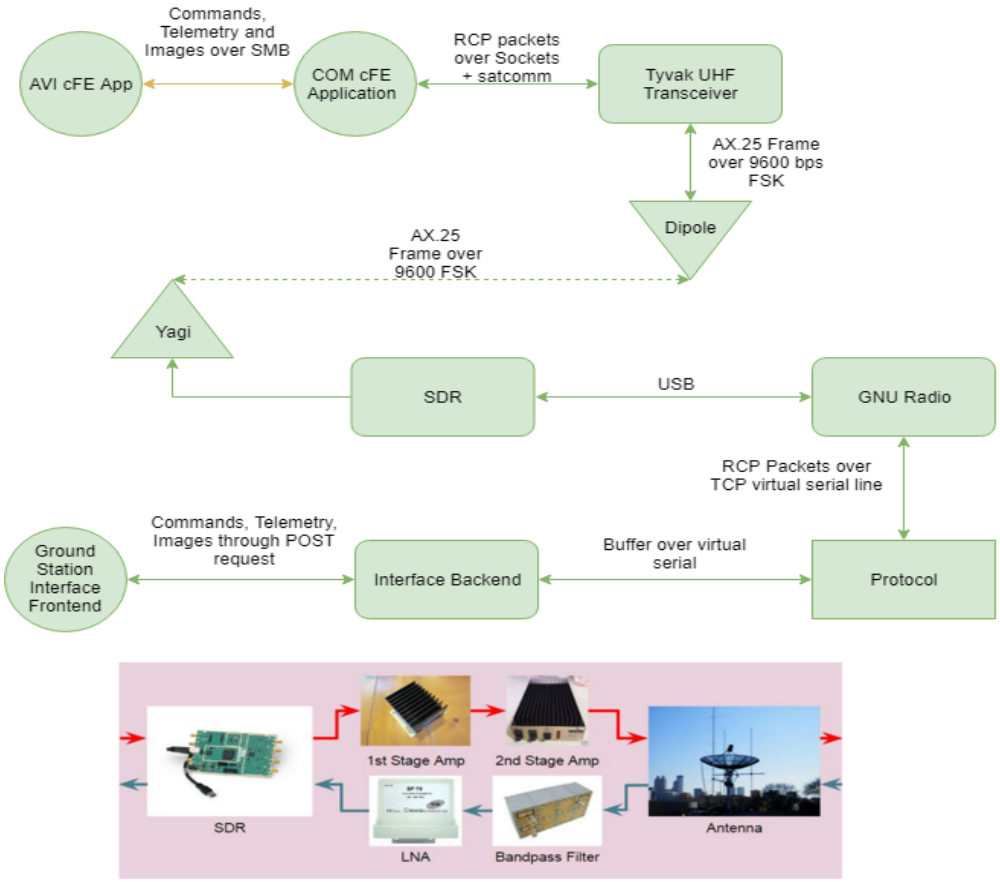

Figure 1. Ground Station Architecture

The ‘kissattach’ and ‘socat’ programs were used to map the SDR to a network interface accessible to the computer using the GNU Radio flowgraph shown in Figures 2 and 3.

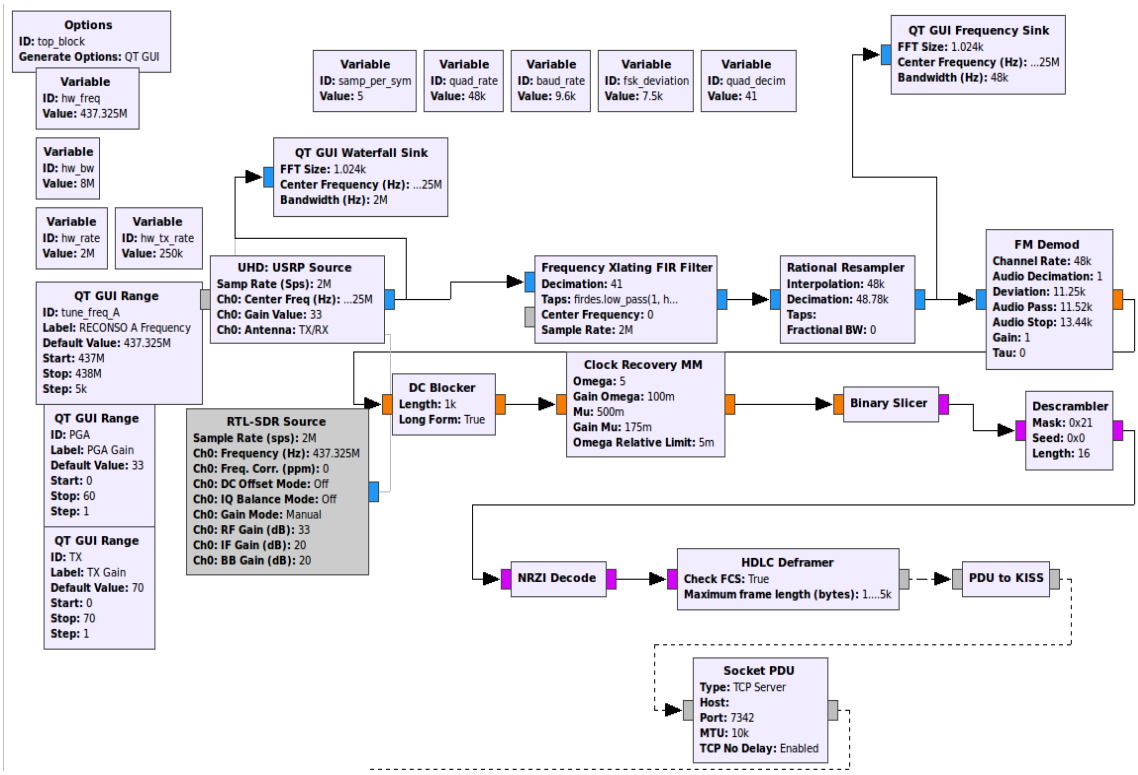

Figure 2. Transmit Flowgraph.

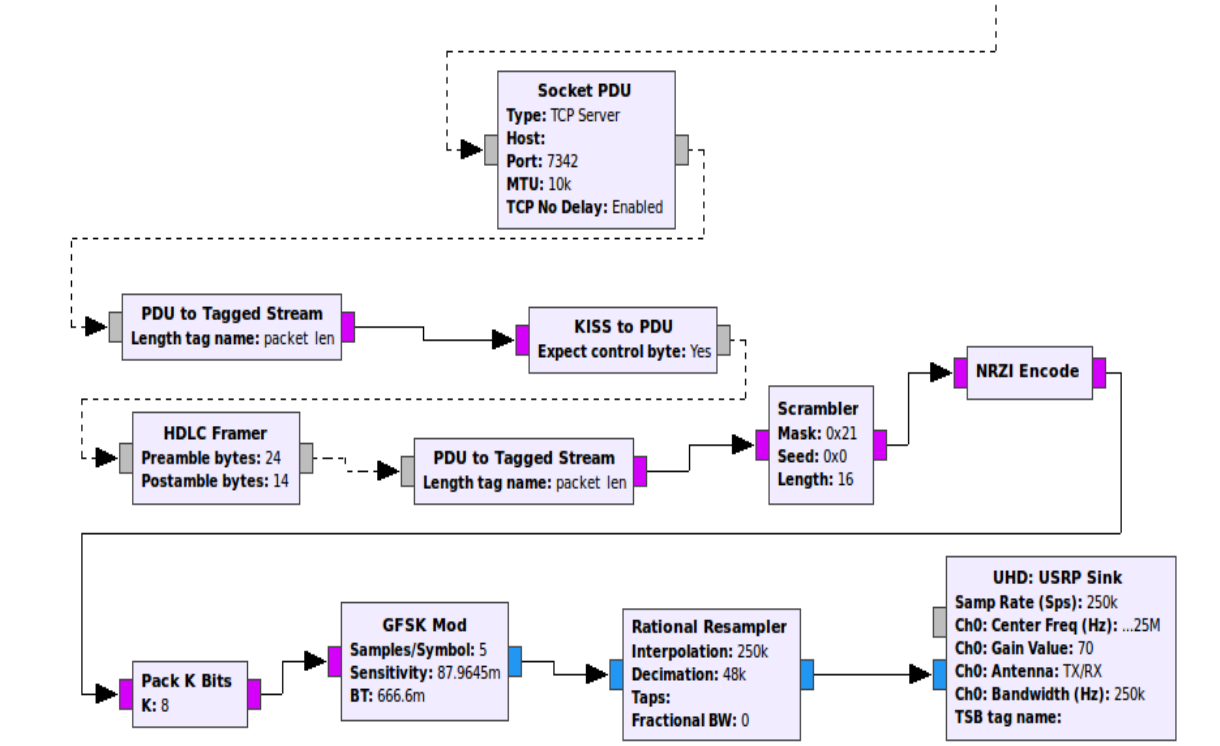

Figure 3. Receive Flowgraph.

Packets sent from the emulated ground station would have a UDP transport layer, IP network layer, and AX.25/KISS link layer.

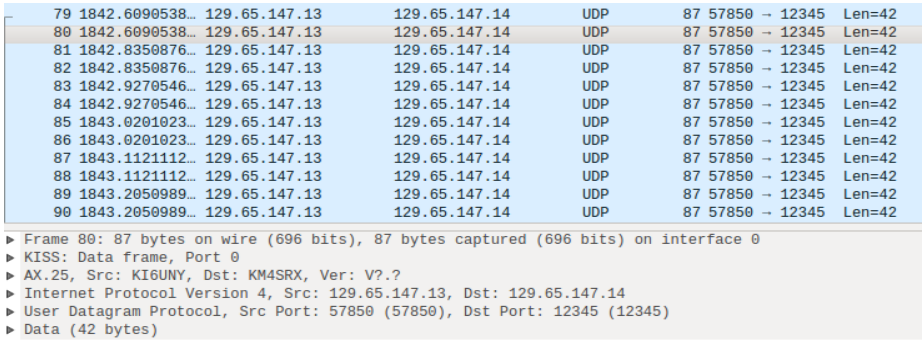

Figure 4. Wireshark capture showing the structure of a packet.

The user interface should allow a user to reliably send commands and have a way to display downlinked results. A preliminary user interface for the ground station is shown below.

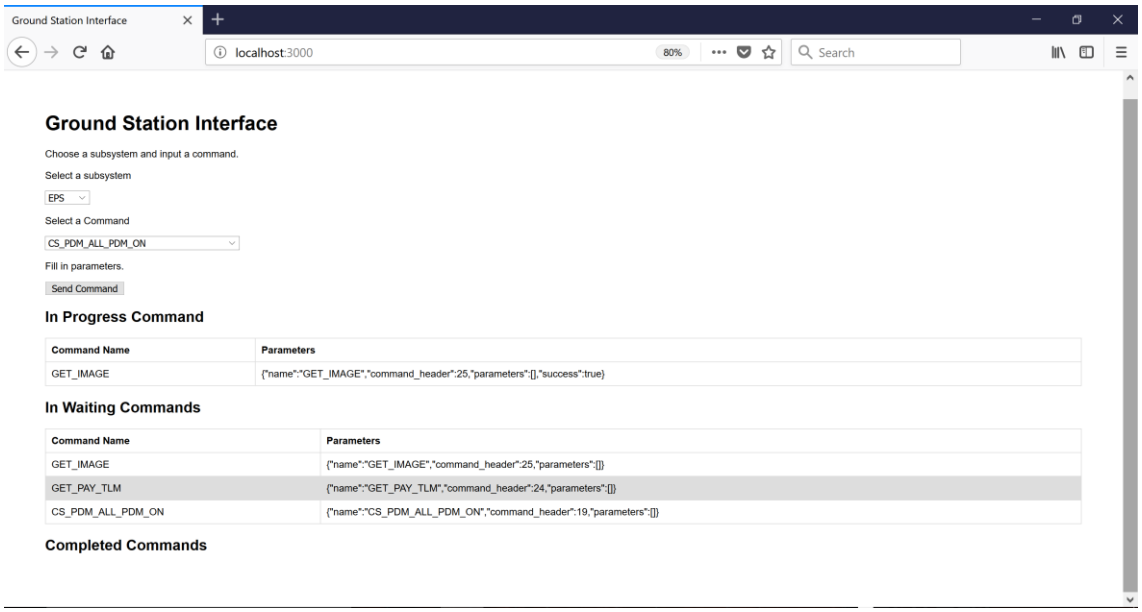

Figure 5. Ground Station Interface.

The ground station user interface is hosted on a Node.js server which communicates information to the transport protocol program via a virtual serial line.

3.2 Satellite Setup

The flight computer consisted of an AntS UHF dipole connected to a UHF radio module on the Tyvak Intrepid processer board, which was attached to an umbilical board that connects to an Ubuntu 15.10 computer USB port via a serial line. The Intrepid board was powered with 4.5 V at a maximum of 2 A.

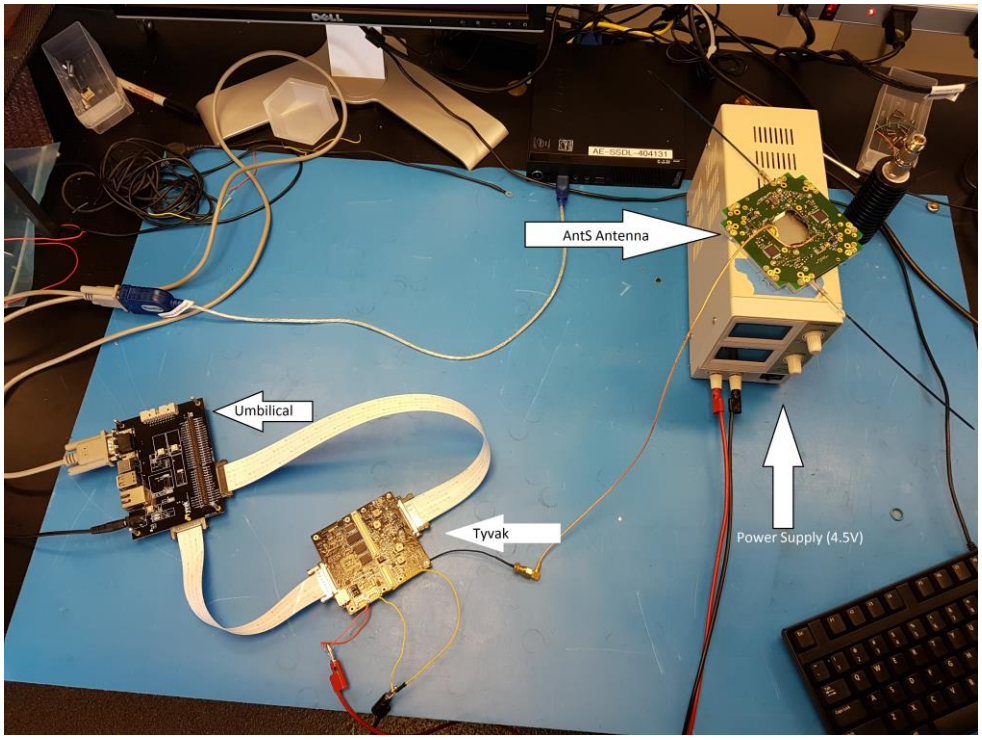

Figure 6. Satellite Setup.

Packets sent from the satellite were identical in structure to the packets shown in Figure 2 from the ground station side.

3.3 Procedures

Different types of protocols were analyzed for transfer time in varying amount of bit error rates and delays. When a protocol was implemented, packets on the receive end were given propagation delays at random for orbits between 500 km - 800 km to match the orbit of the satellite. Each protocol would also have a bit error rate representative of the rate calculated in the link budget for LEO communications. The BER was varied around this range to find the protocol which completed transfers fastest while also maintaining reliably. The information communicated took on both images that were ~5 MB and commands that were ~128 B, which is what is expected from the mission. The protocols that were analyzed included CFDP, SCPS, and CSP. TCP was also run as a benchmark. Captures for each transfer were recorded with Wireshark and analyzed further to determine if parameter tweaking could optimize any one protocol to work better for the mission. The captures were also used in designing new protocols to potentially outperform the ones under testing.

CHAPTER 4

RESULTS

The protocol that is implemented is similar to TCP in determining connection and close states, but it does not have a finish state because timeout periods due to the fact that the connection will timeout when there is no line of sight link between the satellite and ground station. The protocol also has two modes known as the “client mode” and “server mode” due to limitations of simplex connections. The design of the protocol is shown below.

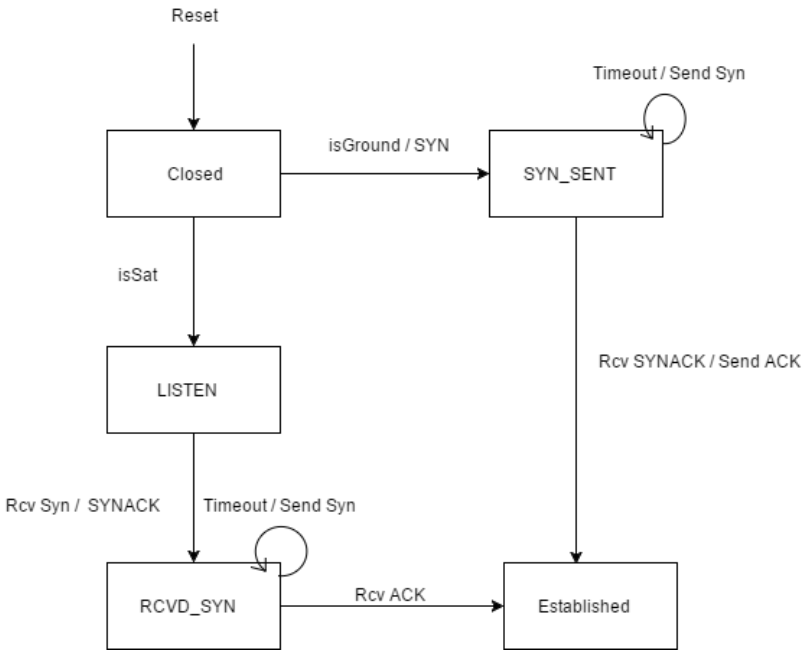

Figure 7. Connection Setup

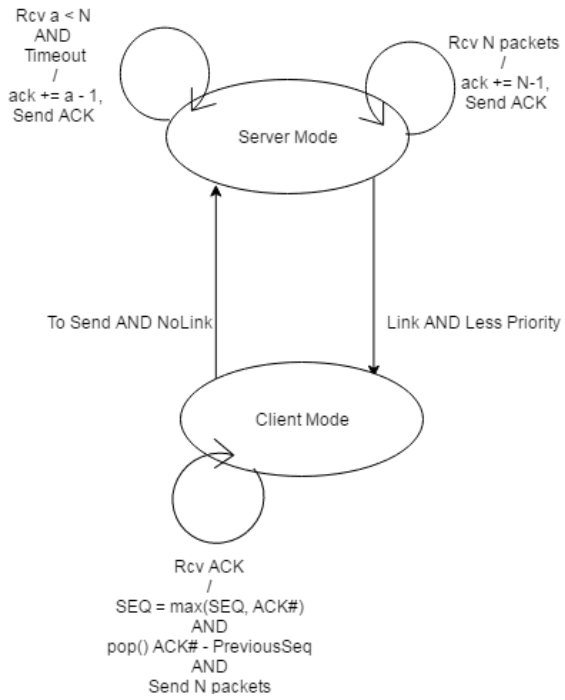

Figure 8. Established loop

The protocol operates similar to that of TCP in that it ACKs received packets, and it does so at intervals of N packets where N is a window size set by the user. The reason that there is no finish state or disconnect, is due to the existence of a timeout. This timeout will be triggered after the satellite is in a part of its orbit with no line of sight to the ground station. There will also be no need for such an operation since there will be only one client (the ground station).

On the 9600 bps link, the protocol is able to transmit a theoretical maximum of 9155 bps of usable data after accounting for overhead with a 1 kB packet size. One of the main tradeoffs for this data rate is the maximum packet size. A graph of this tradeoff is shown below.

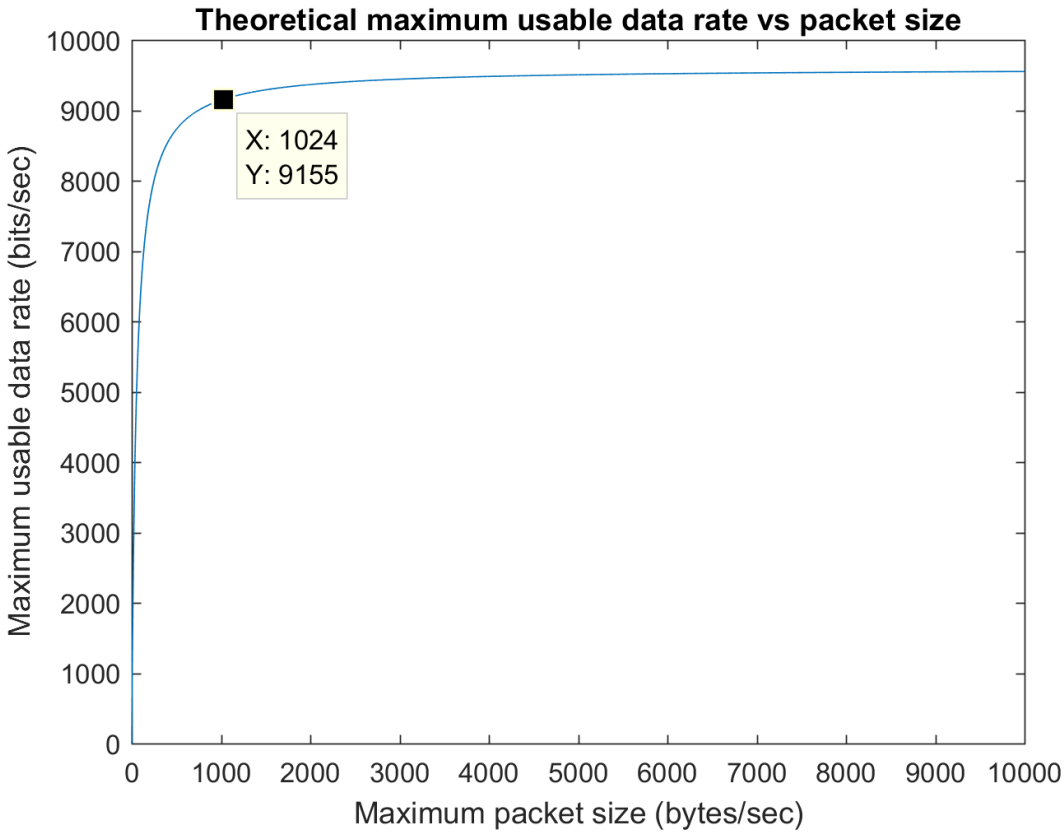

Figure 9. Theoretical maximum usable data rate

As expected, with a large enough packet size, the usable data rate approaches the physical data rate. A large packet size however not realizable due to constraints on packet size in memory, and due to the potential for bit errors. The overhead for this protocol is calculated from the need to keep a TCP header as well as SYN/ACK flags and a sequence number. Additional overhead comes from the overhead imposed in AX.25 packets as well as the preamble and postamble lengths required in the SDR flowgraph shown earlier in the paper. Packet sizes on the order of 1 kB allow for a small enough packet error rate (~ 0.1%) to be negligible due to the small size of commands and image transfers.

The main feature of the protocol, allowing for adjustable switching times, results in better performance over the half duplex RF channel. Based on the requirements of the

hardware, these free parameters can be tunes such that collisions in the RF link are minimized due to switching from transmit mode to receive and vice versa.

From preliminary testing, it is determined that commands for this mission are transmitted on the order of tenths of seconds. The main delay for transfers is due to images. For this mission, although the event only occurs a fixed number of times in the beginning of the mission, images are a critical part of mission success. Images are predicted to take 10 minutes to transfer. This is difficult due to a 12 minute window of access for communication with the ground station. For this reason, chunking image transfers into smaller and more manageable transfers will be explored. Due to the fact that the satellite is still in development, and flight software is likely to change, these timings may also change. A few tests for testing satellite operations still have to be run and more accurate statistics will be able to be taken then.

CHAPTER 5

DISCUSSION

The two modes of the satellite and ground station: “Client” and “Server” are designed specifically for simplex operation. Specifically, uplink from the ground station should always have more priority than downlink from the satellite, therefore there is a path from client mode to server mode to detect if there is a link attempting to be established, and if this link has more priority than what the client is attempting to send. If these conditions are met, the client will be forced into server mode. This method also allows the satellite to generate high priority information, and force the ground station to listen, but this feature is unlikely to be triggered unintentionally if used. This should prevent missions from failing due to links never being able to be established due to long transmit times with large files. Instead, an application protocol can be designed to save interruptions in the connection, and resume after the interruptions are finished.

This type of protocol is also well suited to older transceivers which are still in use today. These transceivers usually have a slow switching time between transmit and receive modes. This means that transmitting and almost instantly receiving a packet at the same time is likely to fail for two reasons: the transceiver is likely to still be in transmit mode, and the connection is a simplex one. The protocol allows the user to specify switching times as parameters, and every switch that has to occur will be adjusted to minimize these types of collisions.

CHAPTER 6

FUTURE WORKS

For future research, the implemented protocol can be optimized for more orbital parameters. For example, if the tumbling data and signal strength is known, the protocol could take advantage of this information to transfer more critical information when a strong signal strength orientation is expected.

Another area where work is needed is in the application layer. Currently, only a transport layer has been designed. This handles link layer obstacles such as transmitter keying and tail delay as well as packet loss, but it does not prioritize information. The application layer can be made to cut off and resume transfers and also order data. This can guarantee that more important information is transferred first.

This protocol should be examined across multiple CubeSat teams to gather more data and compare statistics. It would be interesting to note how the goodput and reliability varies across different environments. For example, a mission which requires a much wider orbit will require higher propagation delays and error rates. In this type of mission, it would be useful to collect statistics on how the protocol compares to others meant for long range missions. It is expected to be not much more useful in more modern setups that do not rely on such legacy equipment because the delay times are almost negligible on newer setups.

APPENDIX A

SOURCE CODE

The source code is freely available on github at: https://github.com/baijund/COM-Protocol .

REFERENCES

[1] Y. Wang, J. Li, G. Chen, and S. Wu, "Repeated sending file delivery protocol in satellite networking communication," *Journal of Systems Engineering and Electronics,* vol. 23, pp. 815-823, 2012.

[2] R. Wang, B. L. Shrestha, X. Wu, T. Wang, A. Ayyagari, E. Tade*, et al.*, "Unreliable CCSDS File Delivery Protocol (CFDP) over Cislunar Communication Links," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 46, pp. 147-169, 2010.

[3] Y. Tak-Shing and E. W. M. Wong, "The scheduled-retransmission multiaccess (SRMA) protocol for packet satellite communication," *IEEE Transactions on Information Theory,* vol. 35, pp. 1319-1324, 1989.

[4] G. Hadjiyiannis, A. Chandrakasan, and S. Devadas, "A low power, low bandwidth protocol for remote wireless terminals," in *Global Telecommunications Conference, 1996. GLOBECOM '96. 'Communications: The Key to Global Prosperity*, 1996, pp. 22-28 vol.1.

[5] A. R. Pujari, N. M. Sawant, S. V. Patil, and P. S. Waghamode, "An effective and efficient approach for low network bandwidth users," in *2015 International Conference on Information Processing (ICIP)*, 2015, pp. 778-783.