Concept network design for a young Mars science station and transplanetary communication

Keely Hill
Computer Science, Florida Polytechnic University, Lakeland, Florida, USA

Abstract - *This paper presents a high-level networking design to provide high bandwidth for a young and growing Mars settlement and science station. A physical network topology is described consisting of a high power ground station to communicate with orbiters. Different parts and devices of the station are connected with network infrastructure. Some examples are given for various non-obvious use cases of the network .*

Additionally, an upper level networking protocol is described to handle reliable communication between planets. This Transplanetary Data Mailing Protocol (TDMP) operates three processes working together: a database of retrievable files and user inbox files; a 'parcel' structure for data to be contained; and parcel piece fragmenting for asynchronous pipelined transmission and loss handling. Together, these systems allow ease of connection between people, scientists, and experiments.

Keywords: Mars networking, interplanetary communication

1 Introduction

This paper presents a high-level networking design for a young and growing Mars settlement and science station of about 20 people; an internet for Mars. The network is intended to provide high-bandwidth for use as the primary network on the station in non-emergency situations. There should be other basic low-gain communication channels for emergency use. This paper also describes a protocol for transplanetary data mailing ("TDMP") of files or data between Earth and Mars where there would be an 8 to 48 minute round-trip-time of any signal depending on the planet's relative positions.

1.1 Habitat size assumptions

A NASA human research program "recommended a minimum acceptable net habitable volume of 25 meters cubed per person."[1] This includes all activities, from sleeping to working to storage. An additional 30 square meter greenhouse, more storage, and workstations results in about 200 square meters of floor space spread across multiple interconnected habitat modules (each with a general purpose).

1.2 Goals and purpose of network and TDMP

As with most networks, the purpose this one is no different. It is to provide connection of computers on the Mars settlement for station logistics (including, but certainly not limited to: climate control, storage inventory tasks, supply management, logging, one time experiments, on going experiments, and connection to nearby external devices). The existing internet protocol version 6 stack is used on the topology to be described. This topology concept is designed to accommodate for the many potential uses of networked connections on a Mars station.

Furthermore, TDMP is to allow reliable, easy to use, automatic communication between Earth and Mars. Examples include: allowing scientists and organizations on earth to connect directly with their experiments, getting intact experimental data or configuring new experiments, reducing the time an astronaut spends on menial tasks; allow pseudo 'face-to-face' contact and important psychological support for the crew from Earth with recorded video messages; and ease communication and coordination of the astronauts and their schedules with Earth.

1.3 Current interplanetary communication[2]

Depending on the planet's orientation at any given time, the Curiosity rover can have (if Earth is in the sky and power is available) anywhere between 500 bits per second to 32,000 bits per second *direct* to earth data transfer rate. Curiosity can transmit an automatically selected data rate of up to two million bits per second to the Mars Reconnaissance Orbiter when it passes overhead. The orbiter could then transmit back to earth over a long period of time at about 3500 bits per second (with obvious lightspeed delay) using software-defined radio.[3]

Orbiters are primarily used for communication back to Earth as they can send much more data due to having more connection time, have more power, and have larger antennae than mobile rovers. The NASA Deep Space Network is used to receive the transmission on Earth. ("Network" is misleading, it refers to the connections between earth based antenna, not connections in space). As construction of a Mars science station begins, more powerful satellites and antenna on both Mars and Earth will be ideal.

2 Topology on the Martian surface

2.1 Satellite orbiters and ground station

In order to provide a communication channel open to earth with the highest frequency, three satellites are to be placed (or use some current satellites) in orbit forming a constellation, maintaining connection to with station at all times. At any time, at least one satellite should have communication with earth (i.e. not be occluded by Mars). As their apparent positions in the sky change, they will have the ability to forward data from the station to an orbiter with an open Earth communication window. Alternatively, a areostationary satellite (akin to Earth's geostationary), could maintain a static apparent position in the sky above the station and forward data accordingly -- this could introduce extra cost, but would be beneficial later for longer distance local Mars communication.

On the surface, there are two (for redundancy) large, high powered ground antennae. These are located near the ground station module. They maintain a more-or-less constant connection to the satellites in orbit. On the rare occurrence, they may also send data directly to earth. The antennae controller interfaces with a computer with large redundant storage drives (along with a small backup computer). This computer operates TDMP described later. In essence, it connects the two planets (or spacecraft). The base station also has an access point for medium range (10 Km) communication.

2.2 Connections

The units of the habitat and various station computers are connected in a mesh network -- like the Internet. Semantically and distance separable units of the station (e.g. crew quarters, general science and work area, greenhouse, etc.) each have their own IPv6 router or switch, connected by Category 7 ethernet cables to multiple neighbors behind wall panels. As the station expands, more connections can be made from leaf routers and switches to the new attached modules or detached habitats.

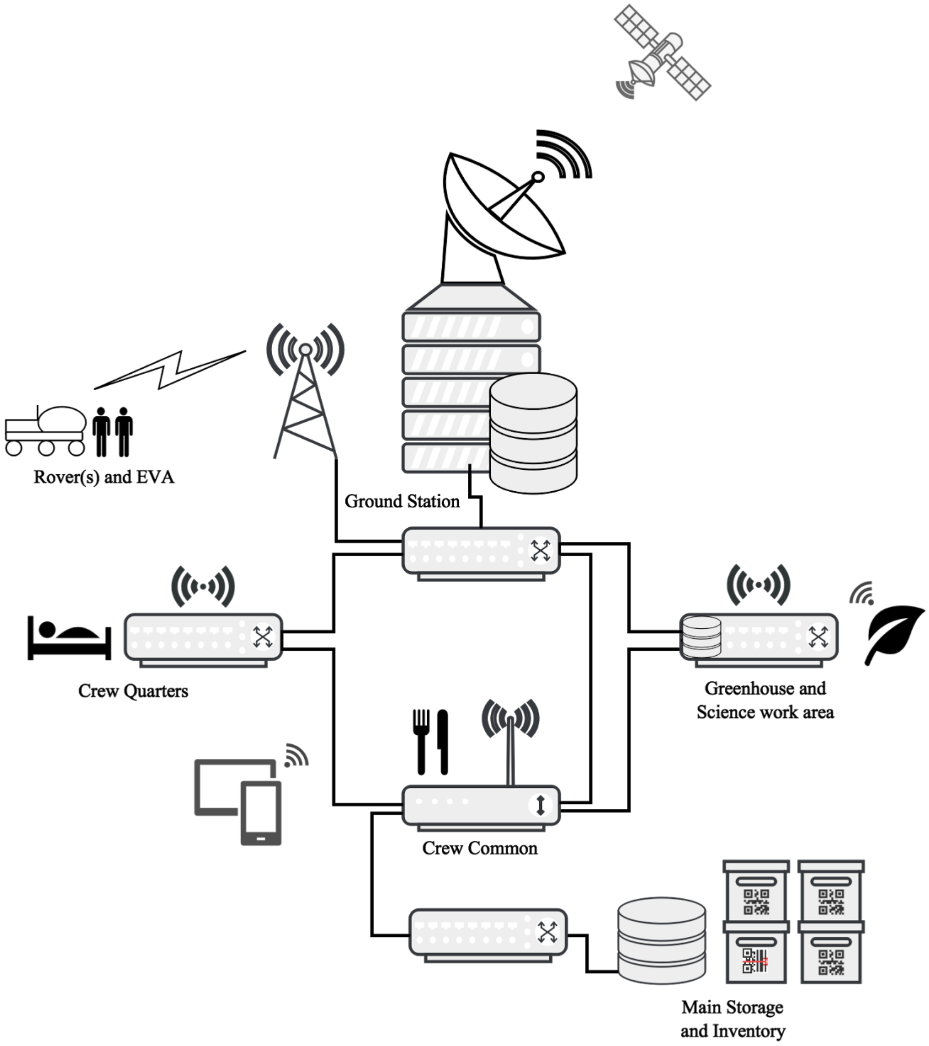


Figure 1. Network Topology Diagram

Personal computers and station devices connect to the network using these routers -- major station servers should have static assigned IP addresses, personal or crew computers may be dynamically assigned. There is an additional, frequency restricted wireless network that connects wireless devices (usually personal crew devices, but may also include sensors) to the network. This 802.11 service is provided by access points from three routers opposite of each other.

Subnets created behind a NAT could also be useful. For example, numerous data collection devices in the greenhouse can have their own access point, connecting them to the greenhouse computer (for earth scientist access) without crowding the main network.

2.3 Other notable connections and examples

The storage and inventory computer is accessible from anywhere on the station, allowing for the important coordination of the usage of supplies (similar to the barcode and computer system operating on the International Space Station today).

Data gathered or live video, for example, could be additionally streamed from an Extravehicular Activity (EVA), or manned or unmanned vehicle mission. Even high quality voice and data communication between EVA partners and the crew members assisting inside is important in doing work outside of a habitat. Unmanned vehicles could even be manually controlled by crew when necessary for a particular task and autonomously send data back.

Small weather (or other scientific) devices could be scattered around and transmit their data directly to the network, then earth. The possibilities are uncountable.

2.4 Potential radiation damage to components

Depending on the depth (underground) and/or the material of the station, radiation protection and occasional part replacement may be needed. Redundant systems ensure data integrity unless the whole system collapses. Computers will fail with time. More detail is out of scope for this paper.

2.5 Topology Diagram

See Figure 1 for a diagram of this conceptual network topology.

3 Transplanetary Data Mailing Protocol

The Transplanetary Data Mailing Protocol (TDMP) is the resulting conceptual protocol from the problem of high latency, unreliable communication between the vast distance of planets. It is a pipelined protocol that combines the metaphors of an inbox, a database, a server, and transport. On the OSI model, it would sit above the network layer and has some combined functionality similar, but not identical, to the session, and transport layers. This description assumes a connection (albeit a poor and long one) between planets that may have some parity check/correction -- lower level than the scope of this paper.

There is a TDMP exists on each planet (or spacecraft). "Local" is used to refer to the network on the planet the TDMP server operates. The files or data are sent as complete "parcels", but broken into pieces for actual transmission.

3.1 Database

A computer running TDMP has a database, it contains long-term files, short-term files, and downloaded files. "Short-term files" are stored only for the purposes of queueing to transmit and potential retransmissions. Once a short-term file has been confirmed as received intact, it is deleted from the sender's database.

Any static resource that can be requested at will, such as movies or music collections, is a "long-term file". It will never be automatically deleted. An index is kept for the long-term files. An index update is transmitted to the other TDMP server weekly should there be changes. The index is a list of metadata about each long-term file: name, hash identifier, and the size.

Finally, a "downloaded file", is a parcel that has been downloaded by the recipient. These have a default lifetime of 1 week for files less than 5GB and 3 days for files larger than 5GB as the server waits for the recipient of the data to retrieve the parcel.

3.2 File identification

Files are identified by their sha512 hash hexadecimal digest. They may also be behind a namespace indicated by a "~". For example: '~movies/304997e6f...b831c4'. This is useful for the organization of the many files. A url to a file on a local TDMP server with an IP of 1.1.1.1 would look like: `tdmp://1.1.1.1/f/{the-hash-id}`.

3.3 Addressing

Both sender and receivers are assigned TDMP addresses. The identifier is not case sensitive. A plus ("+") can be used at the end of the identifier as an automatic alias for filtering rules. The term "user" is used when referring to any human or nonhuman user of the system. For example:

```
[username]@[location name]
```

Mark@mars

PlantTeam.WaterMonitor1@mars

PlantTeam@earth

PlantTeam+input@earth

3.3.1 Reservations

The username "TDMP" is reserved for the computer running the protocol itself.

An asterisk (“*”) username identifier places response data in read only public directory accessible by anyone with a username.

3.4 Local sending and retrieving

This sections explains the format of a “parcel” and how they are used for users to interact with their local TDMP server.

3.4.1 Format of a parcel

A parcel is separated into two parts, the key-value header and the data. The UTF8 encoded header ends when the three byte string `+−+` is alone on a line. The raw or base64 encoded data starts on the following line. See an example parcel of a small test file in the appendix.

The header’s keys are all uppercase, their values start after an equals sign (“=”). TO and FROM are the required addresses of the recipient and sender. There may be multiple recipients separated by a comma. The required key DATALEN, is the base 10 length in bytes of the data. The final required key, HASH, value contains the hexadecimal digest of the sha512 hash of the data.

The following optional headers used for requesting files over space. DATALEN is 0, when requesting files. The header ends normally.

REQUEST: the hexadecimal digest of the files requested (retrieved from the foreign index or in previous personal transmission) including potential namespace. REQTOALL: when “Y” (for yes), the response is to addressed to all..

Other optional headers for general use are: MIME (the mime type of the data), NAME (a file name), ATTACH (a comma separated list hash identifiers of additional files to send that exist in the database), and DATE (Earth time in IETF format).

The “B64” header has three possible values: “N” (for raw data bytes not in base 64), “Y” (for data encoded base64, when the hash is of the decoded data), and “H” (for data encoded in base64, when the hash is of the base64 data)

The “TCPFWD” header, is an IP address on the foreign network and tells the receiving TDMP server to attempt a TCP request from the data of the parcel and return the response. This should be heavily firewalled and filter.

3.4.2 Local parcel retrieval and creation

Once the TDMP computer has verified the successful reception of a parcel, it is then stored for user retrieval. These downloaded files have a limited lifetime on the TDMP server by default (described earlier) and must be dowload elsewhere if they wish to be kept for longer. The ‘frontend’ for the retrieval and sending of parcels over TDMP by users or bots is handled by current internet protocols by the decision of the server.

A secure HTTP server is used to expose a user interface or REST API service for the login, index, inbox, download, parsing, creation, and sending of parcels across interplanetary space (or to local users). Other transport layer protocols (namely TCP and UDP) can be used to facilitate download of various sizes of parcels in addition to the upload and download capabilities of HTTP. An example Python function that creates a parcel from user input is in the appendix section.

Alternatively, a user may configure their account on the TDMP frontend to automatically forward the parcels to another computer for more permanent storage. This same feature may also just be incorporated into the frontend interface.

A TDMP server may also wish to limit the size limit on sending files, storage, and total bandwidth usage based on available transmission time and a user’s needs and tier.

3.5 Handling transmission between planets

Transmission of parcels and their pieces is pipelined and asynchronous. Multiple physical devices may be used to send different pieces from any parcel in queue. The is no built in congestion control, but the physical and technological limits of the transmitting device should be kept in mind.

3.5.1 Parcel pieces

The ability of the receiver to determine the start of a signal at the data link layer is assumed. When a parcel is to be transmitted, first the sha512 hash is taken, then it is broken down into 65 kilobyte (maximum) pieces. Each piece is given an order number (starting at 1) and its data checksummed (as a 16 bit one’s complement sum of the data, order number, and flags). This hash performed is *different* from the hash of the data contained in the parcel, and is used to identify and collect pieces of a parcel by the receiver. The structure of a piece to be transmitted is show below (figure 2).

Octet	0	1
0...15	First 16 bytes of complete parcel hash	
16	Data length (in number of bytes)	
18	Data length (continued)	Flags
20	Order number	
22	Checksum	
24+	Data	

Figure 2.

The ‘flag’ octet uses the first 3 bits for a version number (starting with zero). The next 4 bits are used to indicate: a retransmit request, retransmit cumulative, a complete parcel acknowledgment, and the last piece of a parcel.

This version restricts the piece size to 65 kB due to the current long distant technology bandwidth. The current complete parcel limit is about 4.3GB, which should be enough for a long time over these slow connections. The data length is a 24 bit number, allowing for total parcel sizes greater than 4.3GB, up to about 1TB.

3.5.2 Receiver logic

There are no acknowledgements for individual pieces. The same piece packet structure is used for sending retransmit and parcel acknowledgments. Two issues may occur that would warrant a retransmit request: a failure of checksum or parcel verification and a lost packet indicated by missing pieces when the last piece is received. The piece request is created with a data length of zero, the piece order number to be retransmitted, and the 'retransmit flag' bit being set to one. The 'retransmit cumulative flag' bit (when one) tells the sender to start over sending pieces from the order number of the request. These flags have no affect on a receiver. A timeout timer is used when expecting a retransmission. More pieces may be coming in from the pipeline while all this is occurring.

Once all pieces are successfully received, they are reassembled by the receiver, a final integrity check is performed on the parcel and its contents, then delivered to the inbox of the user and a parcel acknowledgment transmitted to the sender.

4 Note on security

There is no security inherent in TDMP, however, any data can be contained in a parcel allowing for basic signing and encryption of data between parties. PGP, for example, could easily be used on top of TDMP for secure interplanetary communication between individuals. A rudimentary man-in-the middle attack is still possible without previous in-person or verified key exchange -- difficult when 340 million kilometers apart.

At least one trusted party (ideally a semi-independent organization like NASA or the ESA) is needed to provide master signing certificates and start a web-of-trust between humans and computers on each planet. Traditional TLS is difficult, so the hierarchy of certificate authorities or a more decentralized web-of-trust needs to be exchanged and updated in bulk.

5 Lessons learned

Previously to writing this paper, I never thought much about designing a physical network beyond that of a household or small office. I fully explored that via the proposed network topology design.

When first sketching and outlining a design for TDMP, I kept running into branches where I came up with various ideas for performing a specific process or overly 'future proofing' which started to lead to scope-creep. Additionally, making decisions at these various branches would momentarily restrict further design. I quickly realized these issues and was able to better focus in on the problems I was addressing. For example, using assumptions of the capabilities of the networking layers below TDMP.

I also improved my ability to be thinking of implementation while designing TDMP, to then to take what was designed and convert it to small example software. Taking the details, advantages, and disadvantages of existing protocols also help shape the design. Finally of course, more information about long distance communication and error correction with non-earth computers was learned.

6 Conclusion

A network on a Mars station will be needed, and having a reliable high bandwidth one is important. Starting with strong foundation provides for immediate scientific and personal use. It also allows for station growth without network bottlenecking. Reliable communication channels with Earth in a standard high level way lets more and a variety of organizations focus on *their specific work* where transmission of data to and from Mars is the backbone.

7 References

[1] A. Whitmire, L. Leveton, H. Broughton, M. Basner, and A. Kearney, "Minimum Acceptable Net Habitable Volume for Long-Duration Exploration Missions Subject Matter Expert Consensus Session Report." NASA Technical Reports Server, 01-Jan-2014.

[2] "Data Rates/Returns - Mars Science Laboratory," NASA. [Online]. Available: https://mars.nasa.gov/msl/mission/communicationwithearth/data/. [Accessed: 09-Apr-2017].

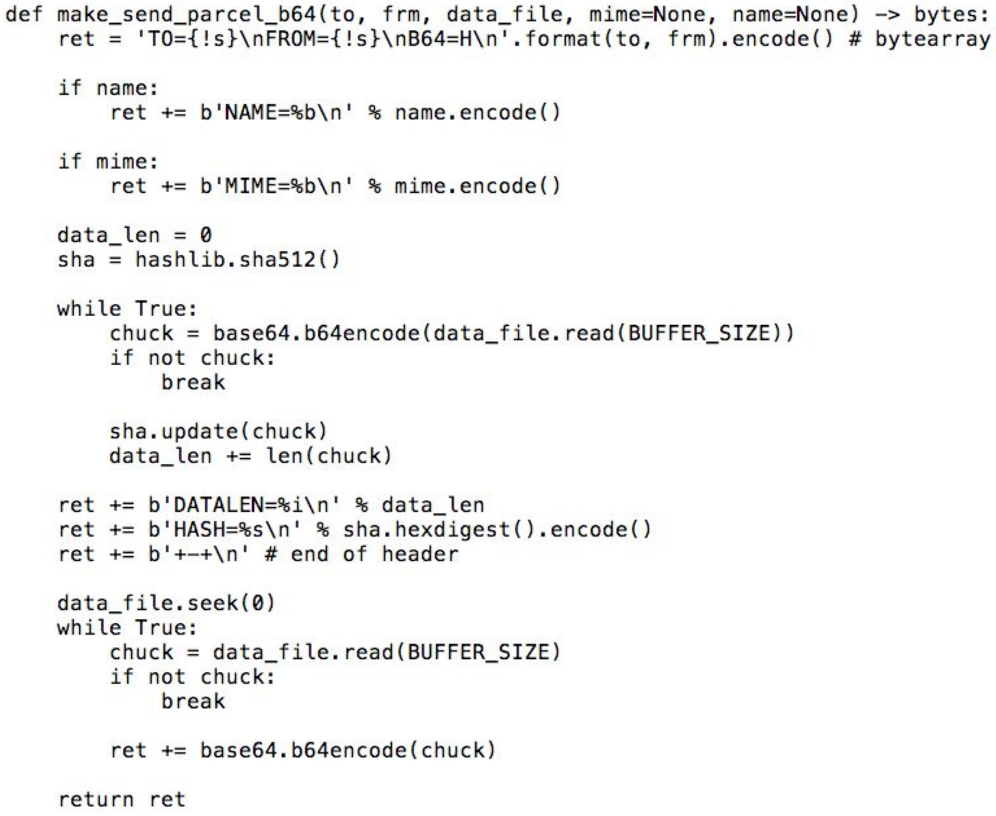
[3] "Mars Reconnaissance Orbiter: The Mission,Spacecraft Parts: Telecommunications" JPL. [Online Archive]. Available: https://web.archive.org/web/20060317102639/http://mars.jpl.nasa.gov/mro/mission/sc_telecomm.html. [Accessed: 09-Apr-2017].

8 Appendix and example code

8.1 An example of a small base64 encoded parcel

```
TO=Steve@mars
FROM=KeelyH@earth
B64=H
NAME=mars-pixel-art.png
MIME=image/png
DATALEN=1476
HASH=786770073e51d19b53f1fa0639ac2477a187b0daddcd4392cef5c7b05aaaf7c11c4e82970c3febe4dc642f111fddbcc084d0a2ef79f0b7
fc2b4651037c5184fd
+-+
iVBORw0KGgoAAAANSUhEUgAAAB4AAAAeCAYAAAA7MK6iAAAABGdBTUEAALGPC/xhBQAAACBjSFJNAAB6JgAAgIQAAPoAAACA6AAAdTAAAOpgAAA6mAA
AF3CculE8AAABWWlUWHRYTUw6Y29tLmFkb2JlLnhtcAAAAAAAPHg6eG1wbWV0YSB4bWxuczp4PSJhZG9iZTpuczptZXRhLyIgeDp4bXB0az0iWE1QIE
NvcmUgNS40LjAiPgogICA8cmRmOlJERiB4bWxuczpyZGY9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkvMDIvMjItcmRmLXN5bnRheC1ucyMiPgogICAgI
CA8cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0iIgogICAgICAgICAgICB4bWxuczp0aWZmPSJodHRwOi8vbnMuYWRvYmUuY29tL3RpZmYvMS4wLyI+
CiAgICAgICAgIDx0aWZmOk9yaWVudGF0aW9uPjE8L3RpZmY6T3JpZW50YXRpb24+CiAgICAgIDwvcmRmOkRlc2NyaXB0aW9uPgogICA8L3JkZjpSREY
+CjwveDp4bXBtZXRhPgpMwidZAAACeUlEQVRIDaWWvW5TQRCF15ZlRaLJnwKiokHuQDwGFc9KxXNAl44qSiInSoMUoSjmfpd8l7njvT8hKzmzs3N2zs
6ZXTuL8ozxdbXajcG/PDwsxuIxNgmcIovJ4nzqEMsIzvP/JSVPI82oOqtMhv8SQvP9OD0tu+22JW9k3VN2j/glpOvj4/L79ra8OzwsP+/uSmnIGRwgk
/dOMpcUgqHx9vGxXCz/dtBDgP2w3VJ2xzfa46HktXVI8ugqbwJIH0cn9dxq3VwjIobEURHIHVw4q25Lz6TNU2ixzbp7imvfzs66tc/X1yX7BF0j7vj+
VLGSD0oNqWTYeAiSDVUMaSTEl5R9V5sNpgwSt9Hmj+T6EPJhnYSQDB3CPTW7zDJHkFVqicWeifUZ6U/Zy81mN1nxVBLjXqha3z82T61914Ib+2xiEiA
zvSNhraf2mBgfBkr5vvEniZVZia2MzSZlPmfw5eL4915cebKxb/m0CTroWnkEXN3clNcnJ2WRL1esiNtaI2U94mLiPLdKWqRqEPekNll8HvQFP66Jyy
TZl5QckorpSW3yOYnB1nCSSYCtrVW/MmsJY6KheY0gY5H5zfn5oid1BuFnmSNGIqxzeom0rw4OInRv3iOmUqudI3t8l2bOvaxhwHY/zN7uTGxC1q3KN
ZN6SEmt9tf9vdD2CeEgM7YjxoG8RuwamKnxfr3uIBBzCKy9Ndi71ZHA0wO8ED3TxkolzVt7FRNsvgbb/wyRVSmHJM7J9COeShlKLGaPmIDkgqKNSVn3
cGJifIgUbJXYJB7AZJlEXLbgc08zZpQYsOR5o/6noyOnPZul7QUbZ5I4bsiHyKRTZDHXHxe/Jp5RiDkVAAAAAElFTkSuQmCC
```

8.2 Simple function making a TDMP parcel with base64 encoded data

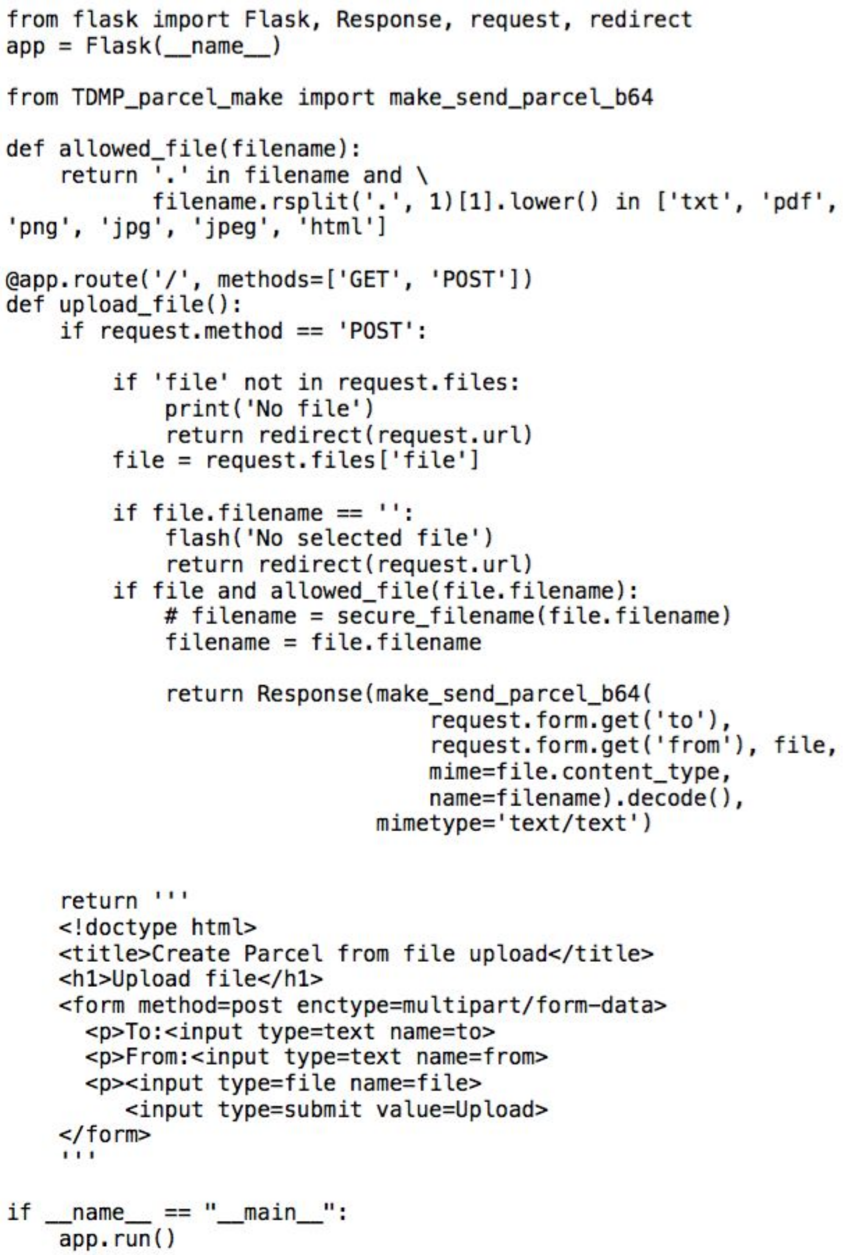


```python
def make_send_parcel_b64(to, frm, data_file, mime=None, name=None) -> bytes:
    ret = 'TO={!s}\nFROM={!s}\nB64=H\n'.format(to, frm).encode() # bytearray

    if name:
        ret += b'NAME=%b\n' % name.encode()

    if mime:
        ret += b'MIME=%b\n' % mime.encode()

    data_len = 0
    sha = hashlib.sha512()

    while True:
        chuck = base64.b64encode(data_file.read(BUFFER_SIZE))
        if not chuck:
            break

        sha.update(chuck)
        data_len += len(chuck)

    ret += b'DATALEN=%i\n' % data_len
    ret += b'HASH=%s\n' % sha.hexdigest().encode()
    ret += b'+-+\n' # end of header

    data_file.seek(0)
    while True:
        chuck = data_file.read(BUFFER_SIZE)
        if not chuck:
            break

        ret += base64.b64encode(chuck)

    return ret
```

Note BUFFER_SIZE must be divisible by 3; e.g. 32760.

8.3 Example TDMP frontend

A simple Flask web app acting as a frontend, taking user input, and outputting the parcel using the previous function for parcel creation.



```
from flask import Flask, Response, request, redirect
app = Flask(__name__)

from TDMP_parcel_make import make_send_parcel_b64

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ['txt', 'pdf',
'png', 'jpg', 'jpeg', 'html']

@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':

        if 'file' not in request.files:
            print('No file')
            return redirect(request.url)
        file = request.files['file']

        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            # filename = secure_filename(file.filename)
            filename = file.filename

            return Response(make_send_parcel_b64(
                                request.form.get('to'),
                                request.form.get('from'), file,
                                mime=file.content_type,
                                name=filename).decode(),
                            mimetype='text/text')


    return '''
    <!doctype html>
    <title>Create Parcel from file upload</title>
    <h1>Upload file</h1>
    <form method=post enctype=multipart/form-data>
      <p>To:<input type=text name=to>
      <p>From:<input type=text name=from>
      <p><input type=file name=file>
         <input type=submit value=Upload>
    </form>
    '''

if __name__ == "__main__":
    app.run()
```