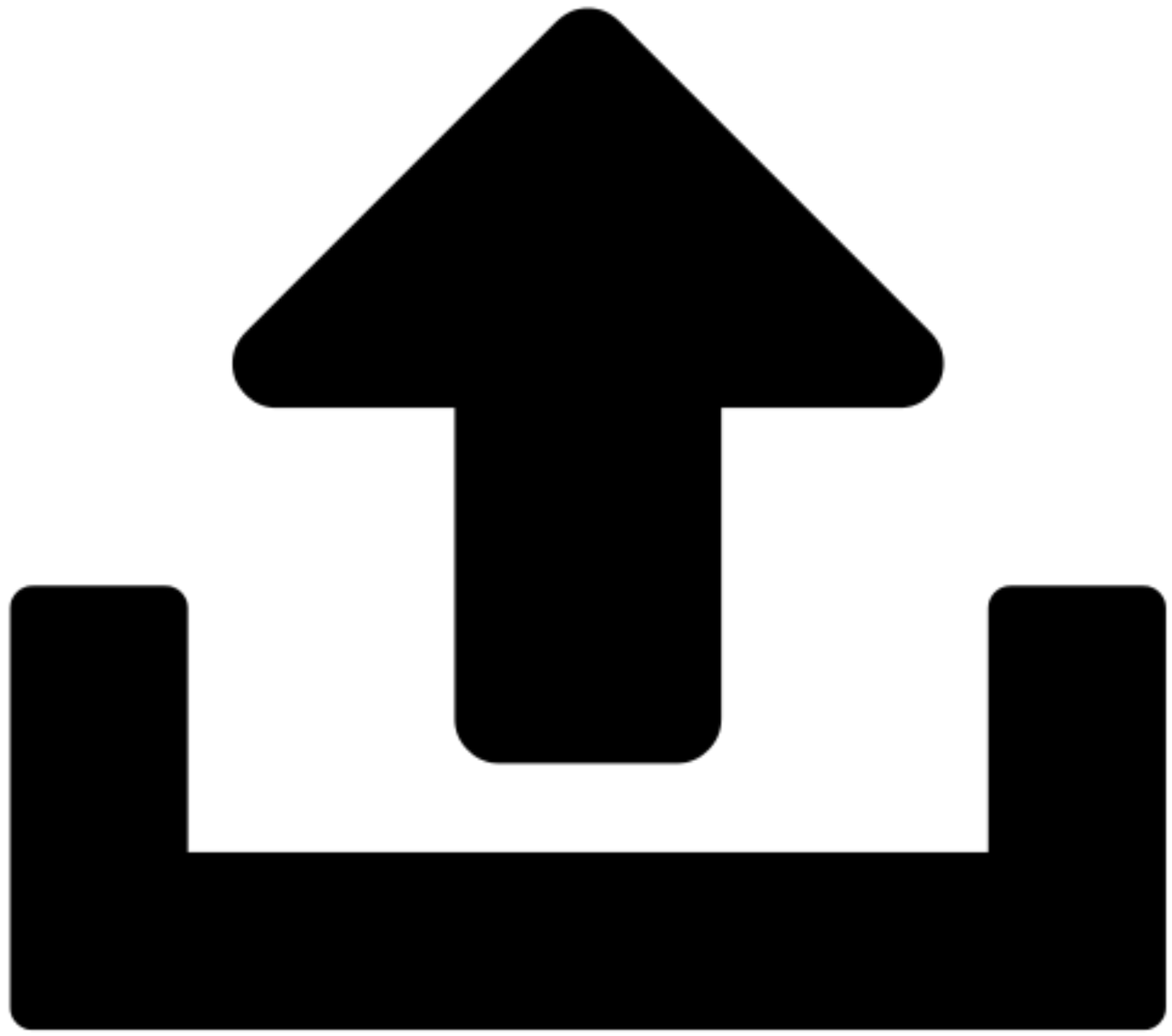
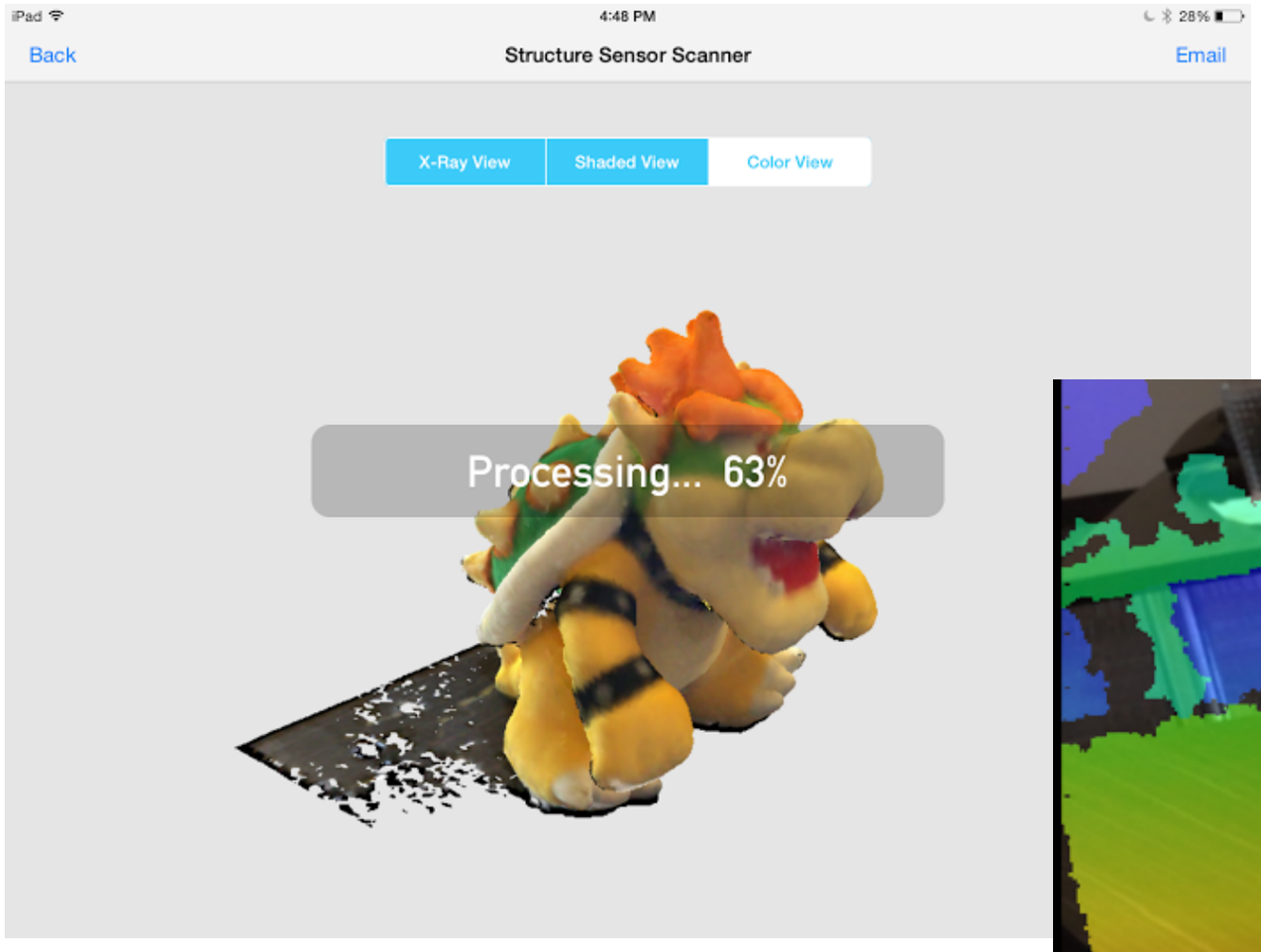
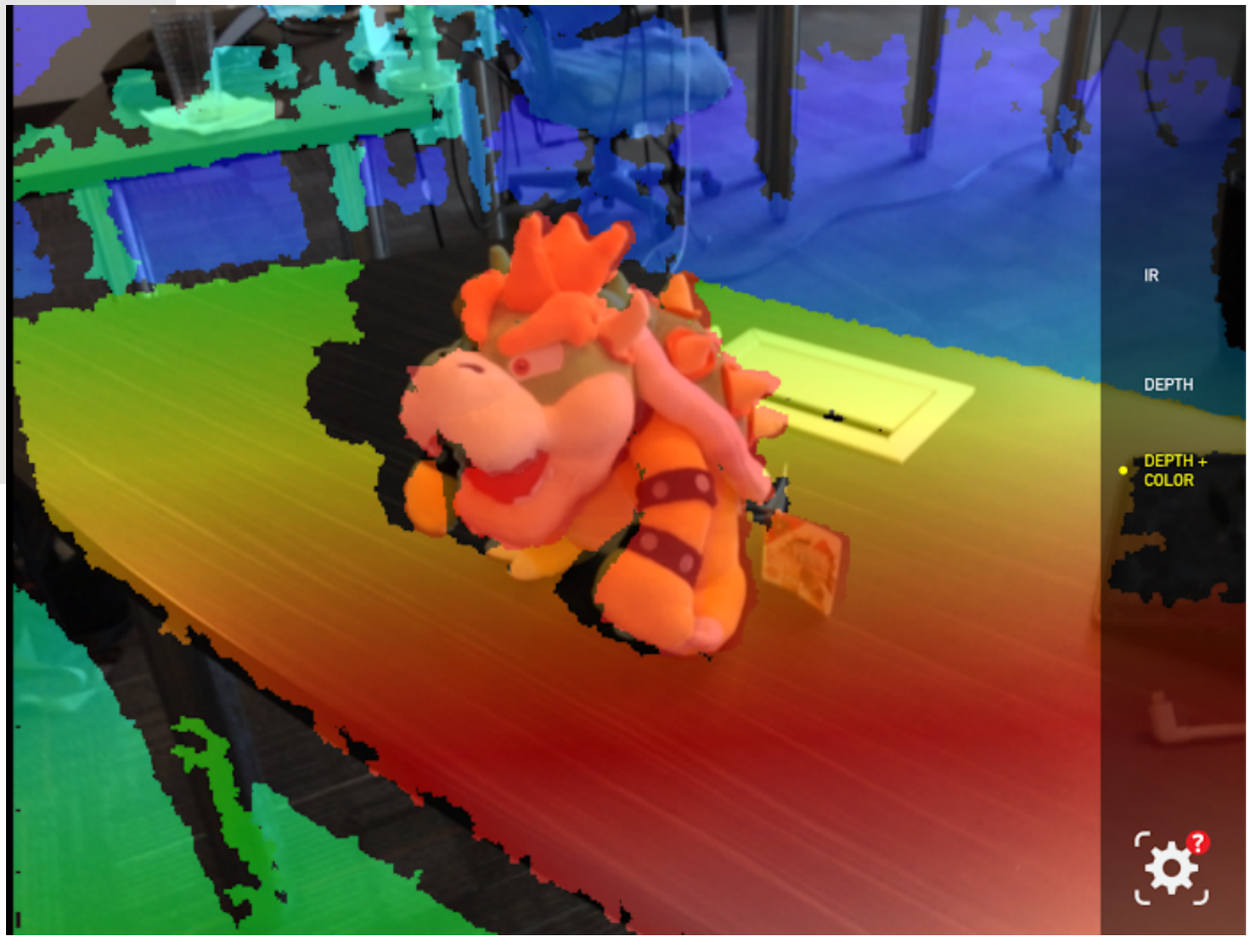
Methods for File Uploading



Greg Schafer
2015/05/26

Motivation


iPad
4:48 PM
28%
Back
Structure Sensor Scanner
Email
X-Ray View
Shaded View
Color View
Processing... 63%

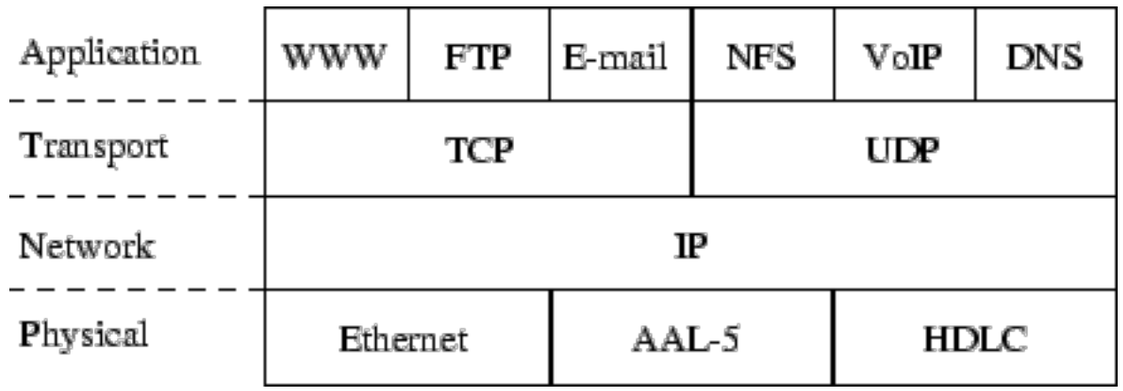

IR
DEPTH
DEPTH +
COLOR

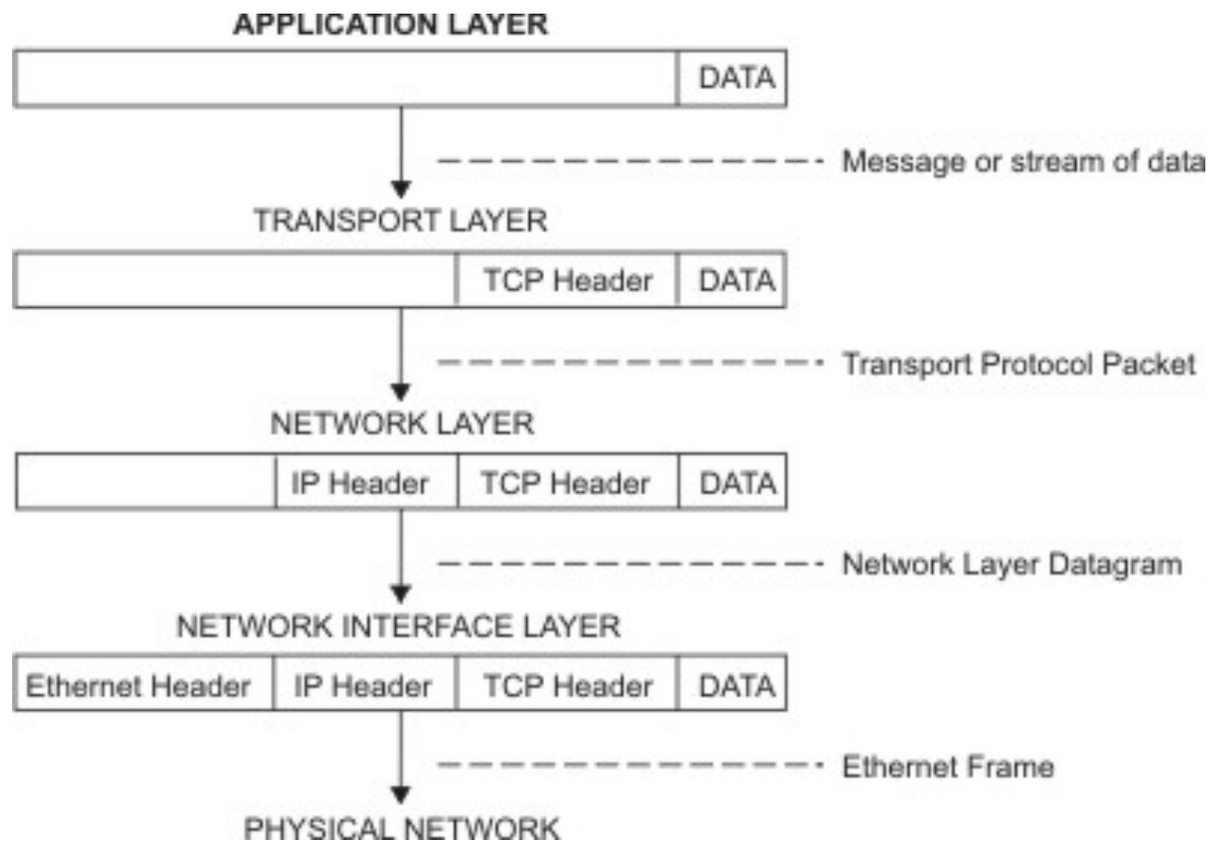
Roadmap

- Low level
  - Bytes and Packets / Networking
  - Request Structure (Multipart, RFCs)
- Basic uploader
- Chunk uploader (a la S3)

TCP

- Used for WWW, smtp, ftp, ssh, etc.
- Reliable transport
- Provides ordering
- Provides retransmissions (acks)
- Accurate delivery instead of timely delivery

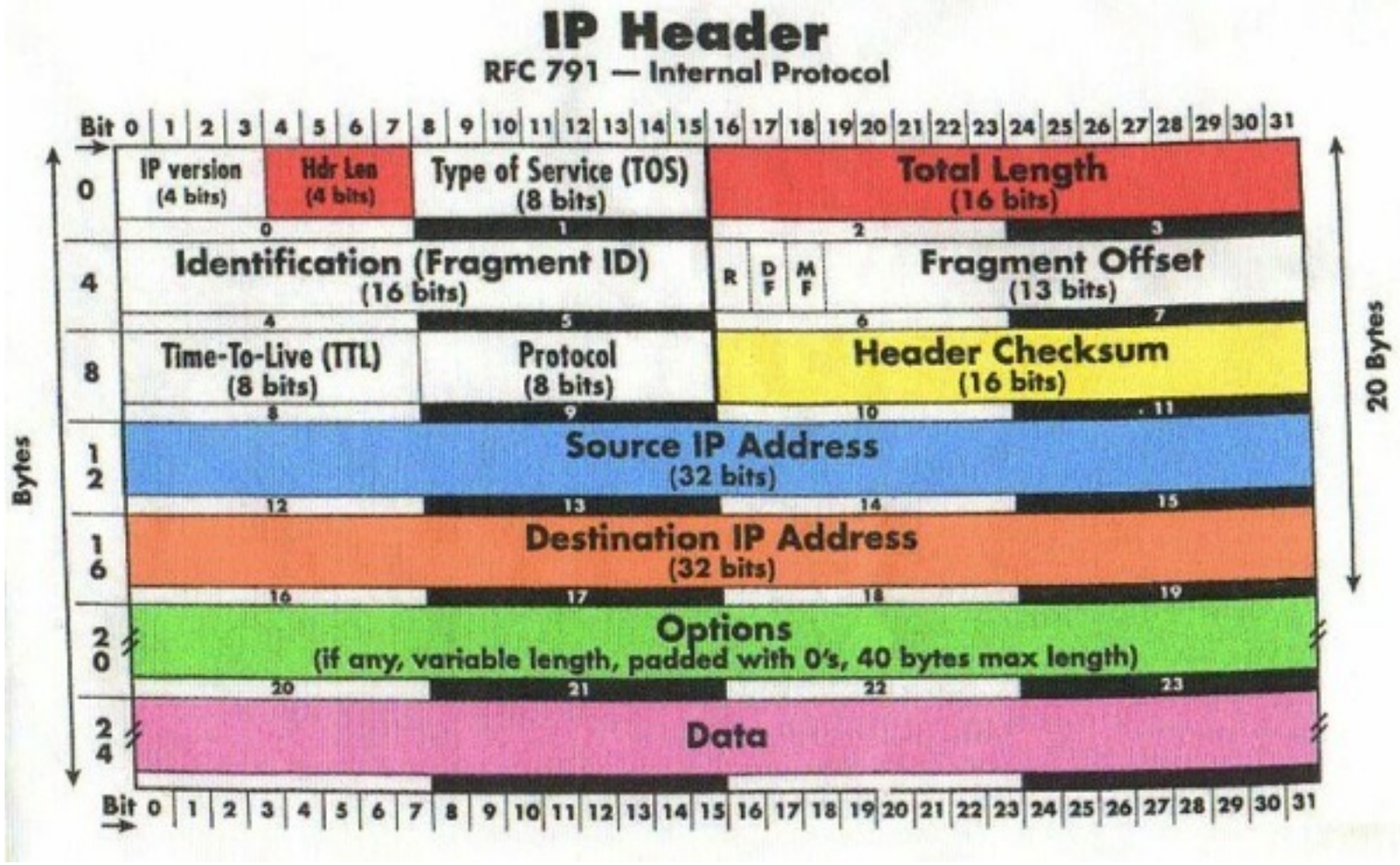




TCP

- sudo tcpdump -X -i eth0 'dst host 10.1.10.242 and dst port 8000'

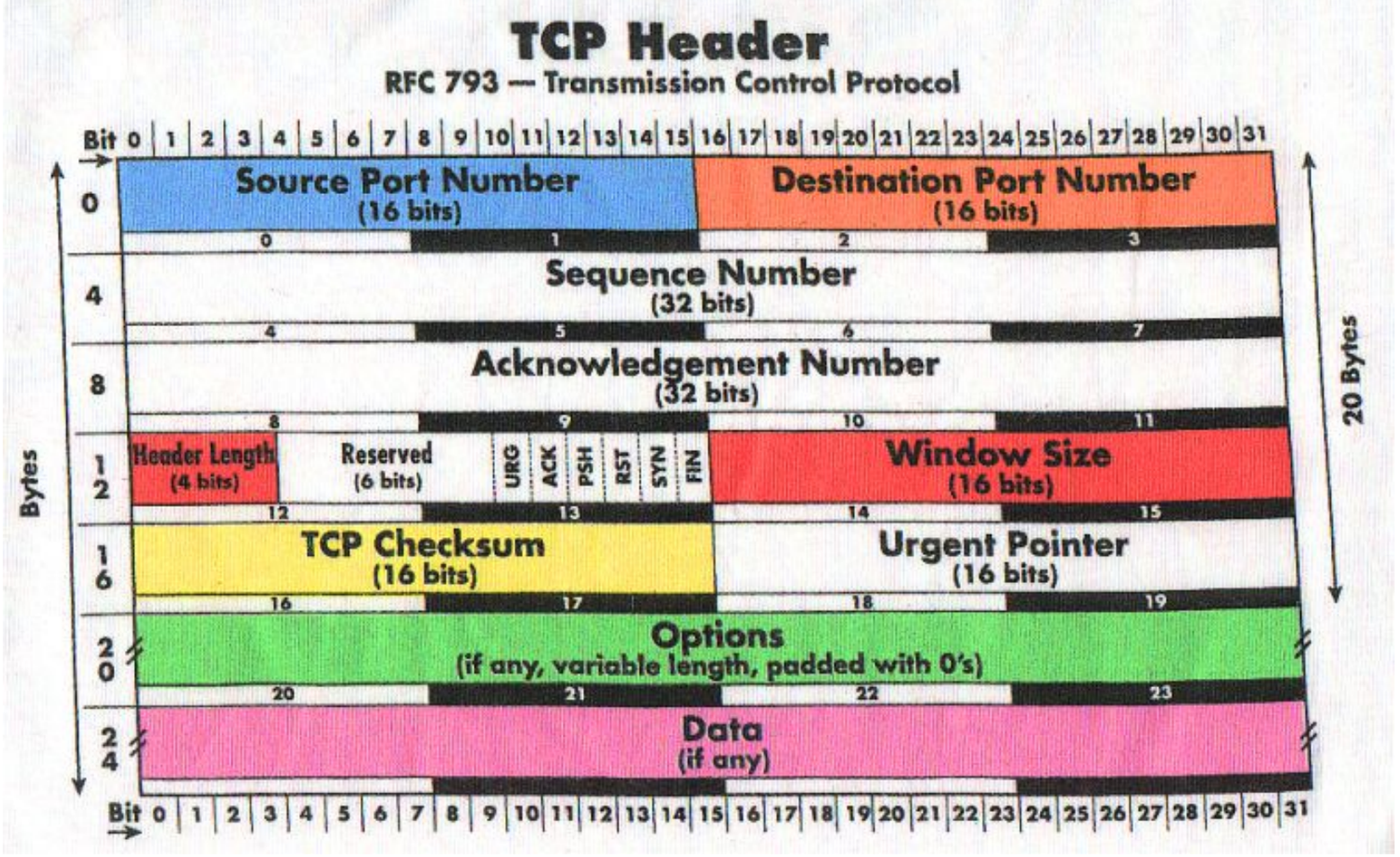
```
17:59:10.338780 IP 10.1.10.15.57345 > 10.1.10.242.8000: Flags [.], ack 4127002078,
        0x0000:  4500 0034 bb37 4000 4006 568a 0a01 0a0f  E..4.7@.@.V.....
        0x0010:  0a01 0af2 e001 1f40 8a9c ca03 f5fd 0dde  .......@........
        0x0020:  8010 1015 f679 0000 0101 080a 1b3d 57f4  .....y.......=W.
        0x0030:  009a 7ba2                                ..{.
```



TCP

- sudo tcpdump -X -i eth0 'dst host 10.1.10.242 and dst port 8000'

```
17:59:10.338780 IP 10.1.10.15.57345 > 10.1.10.242.8000: Flags [.], ack 4127002078,
IP Header
        0x0000:  4500 0034 bb37 4000 4006 568a 0a01 0a0f  E..4.7@.@.V.....
        0x0010:  0a01 0af2 e001 1f40 8a9c ca03 f5fd 0dde  .......@........
        0x0020:  8010 1015 f679 0000 0101 080a 1b3d 57f4  .....y.......=W.
        0x0030:  009a 7ba2                                ..{.
```



Request Structure of an Upload

- Browser Inspector
  - Chrome
  - Firefox

▼ Request Payload

```
------WebKitFormBoundaryHgF2HhPOd9YHTxHM
Content-Disposition: form-data; name="csrfmiddlewaretoken"

wVPCOSlsChpjJTQXV42AhBDWAo4eYaY4
------WebKitFormBoundaryHgF2HhPOd9YHTxHM
Content-Disposition: form-data; name="file"; filename="_uploadme.txt"
Content-Type: text/plain


------WebKitFormBoundaryHgF2HhPOd9YHTxHM
Content-Disposition: form-data; name="file2"; filename="_uploadme"
Content-Type: application/octet-stream

------WebKitFormBoundaryHgF2HhPOd9YHTxHM--
```

▼ Request payload

```
1  Content-Type: multipart/form-data; boundary=----------------------------167449251742253928420368092 06
2  Content-Length: 595
3
4  ------------------------------167449251742253928420368092 06
5  Content-Disposition: form-data; name="csrfmiddlewaretoken"
6
7  fsDZiLdlbbtwlLT8fQeMO4x0PhpGAxyy
8  ------------------------------167449251742253928420368092 06
9  Content-Disposition: form-data; name="file"; filename="_uploadme.txt"
10 Content-Type: text/plain
11
12 Hello World
13
14 ------------------------------167449251742253928420368092 06
15 Content-Disposition: form-data; name="file2"; filename="_uploadme"
16 Content-Type: application/octet-stream
17
18 ¢£¤¥¦§¨©ª²³´µ¶·¸¹ºÂÃÄÅÆÇÈÉÊ
19 ------------------------------167449251742253928420368092 06--
20
```

Request Structure of an Upload

- Uploading via a form
  - Content-Types
  - multipart/form-data RFC

4.5 Charset of text in form data

Each part of a multipart/form-data is supposed to have a content-type. In the case where a field element is text, the charset parameter for the text indicates the character encoding used.

For example, a form with a text field in which a user typed 'Joe owes <eu>100' where <eu> is the Euro symbol might have form data returned as:

```
--AaB03x
content-disposition: form-data; name="field1"
content-type: text/plain;charset=windows-1250
content-transfer-encoding: quoted-printable
```


- Uploading via API
  - You can do whatever you want as long as both sides speak same language

Basic Uploader

- With and without FileField

Basic Uploader

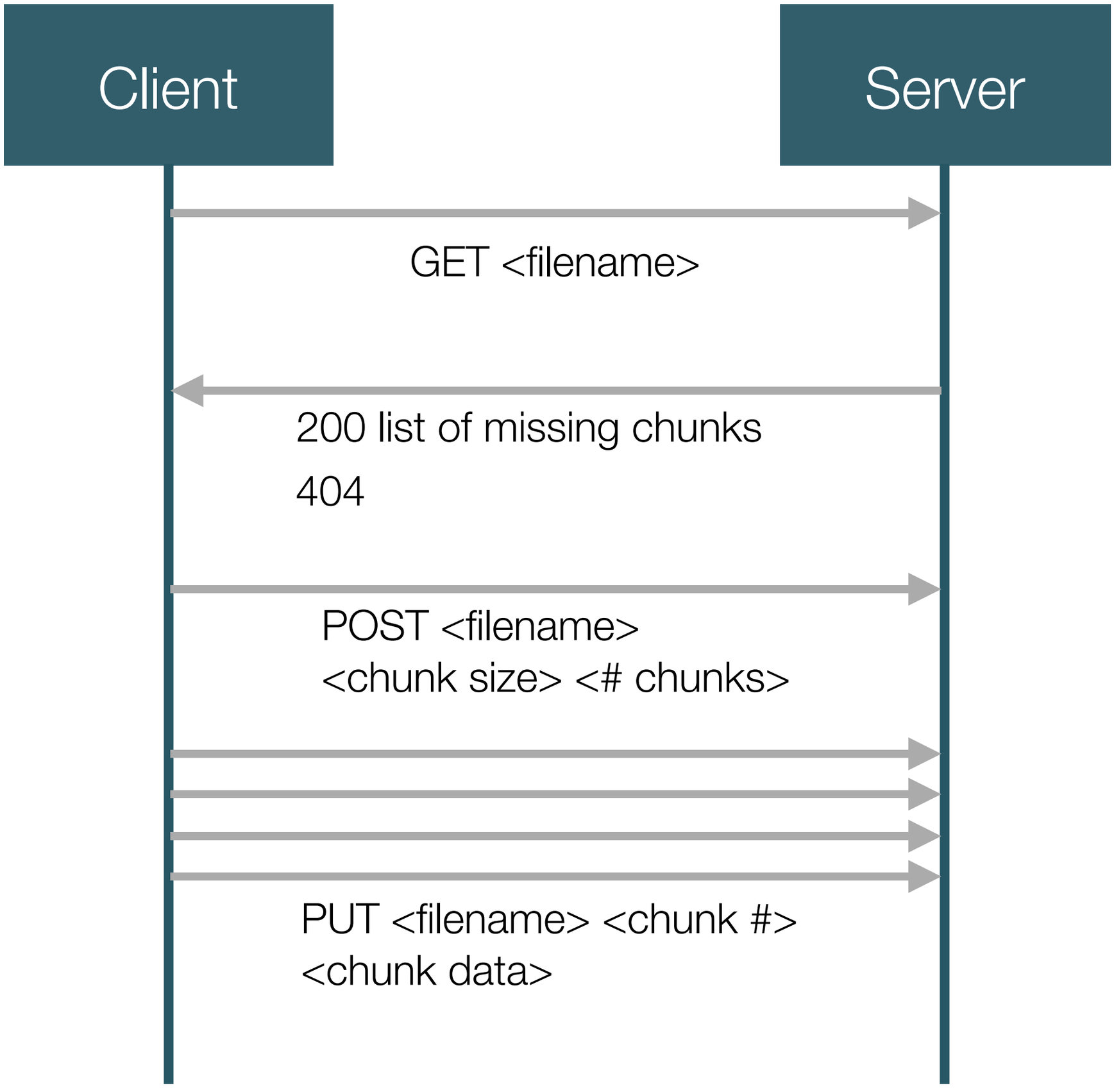
- Two Django upload handlers:
  - MemoryFileUploadHandler
    - used by default for files < 2.5 MB
  - TemporaryFileUploadHandler
    - use file.chunks() for inline processing
- Django upload settings

Chunk Uploader

- Why use chunked uploader?
    - Resumable uploads for intermittent/lossy connections (e.g. mobile data plan) and relatively large upload

- Google Drive API
- AWS S3 API
- Browser-side Javascript APIs

Chunk uploader



Consideration

- Basic uploader better for
  - Reliable internet connection
  - Code simplicity
  - Small files (less overhead)
  - Statelessness (don’t need sticky load-balancer)
- Chunk uploader better for
  - Lossy connections (mobile, data plans)
  - High latency connections
  - Large files

Next Steps

- Measure performance, throughput
  - How to measure in isolation? (disk i/o)
  - The effect of:
    - File size
    - Chunk size
    - Channel lossy-ness
    - Channel latency
- on:
  - Throughput/time taken
  - Extra data sent

Thanks!