Multisource and Multipath Content Transfer

R & D Project Presentation

Shruti Hiray

Guide: Prof. D. Manjunath

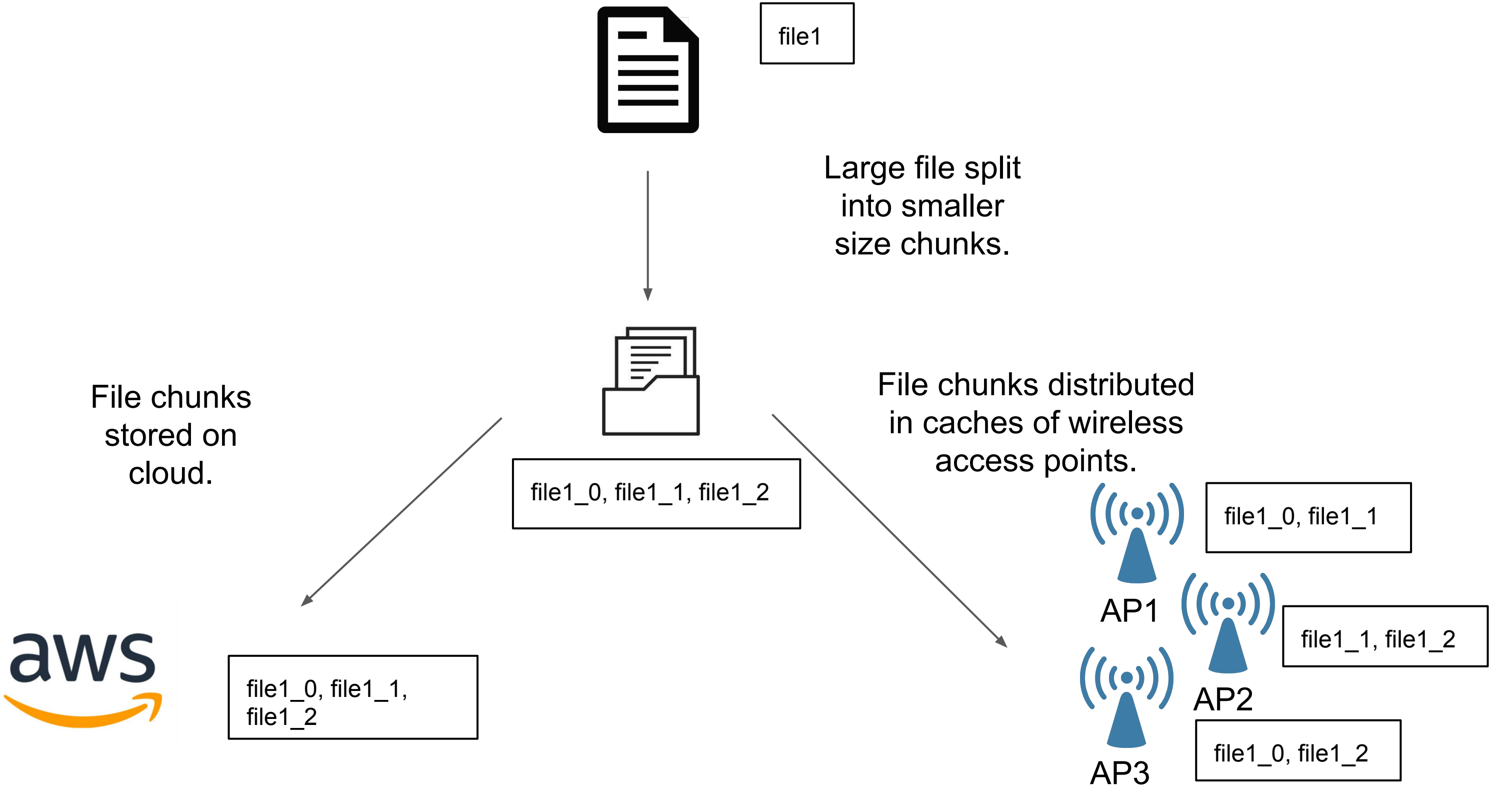
Department of Electrical Engineering
IIT Bombay

November 2018

Introduction

- Devices can make use of multiple network interface (WiFi, Ethernet, Cellular) to their advantage by downloading a file through multipath.
- Parallel download can increase overall throughput of file download.
- Files can be broken into smaller chunks, some chunks of which are stored locally. For example, in cache of wireless access points.
- Obtaining part of file from local cache of a wireless access point:
    - Decreases the path length of data transfer,
    - Leads to lesser download time and less network congestion.
- Clients can thus obtain parts of file from multiple sources (Access Point, Remote File Server) through multiple paths.

Scheme

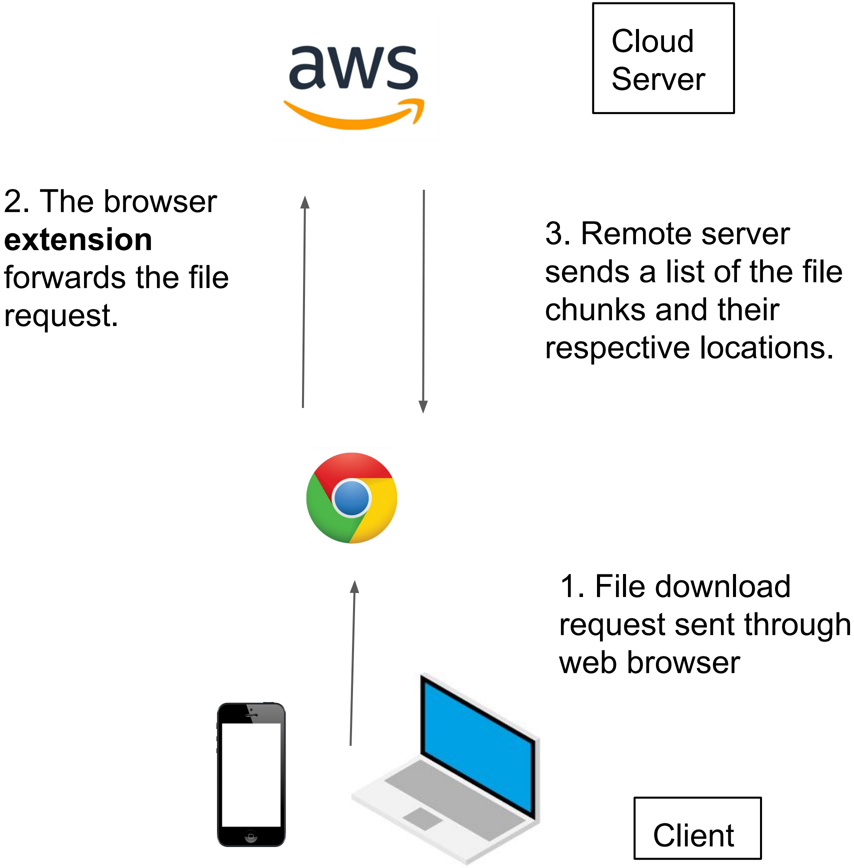


Large file split into smaller size chunks.

File chunks stored on cloud.

File chunks distributed in caches of wireless access points.

Scheme (continued)



Cloud Server

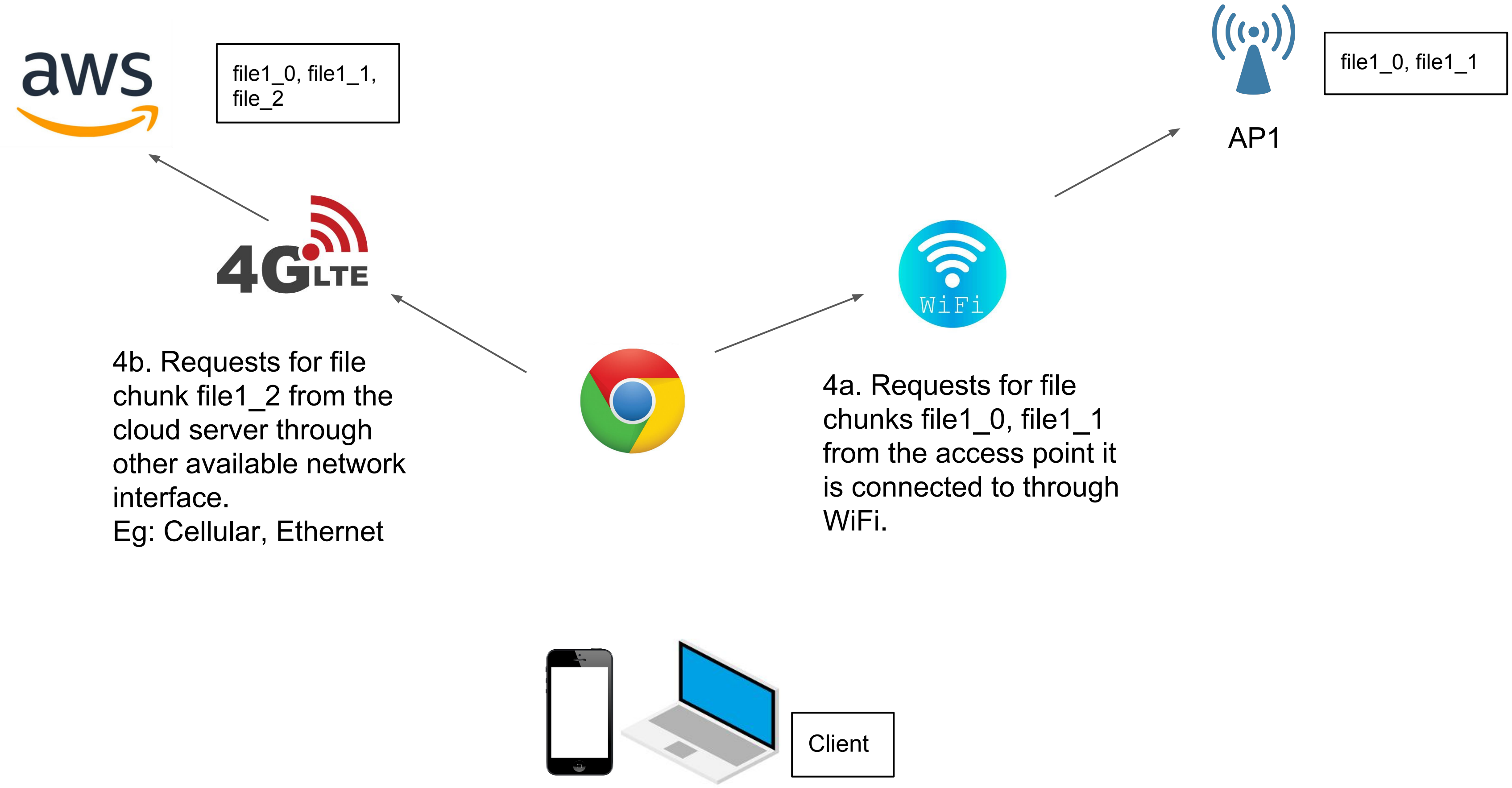
2. The browser **extension** forwards the file request.

3. Remote server sends a list of the file chunks and their respective locations.

1. File download request sent through web browser

Client

Scheme (continued)



file1_0, file1_1, file_2

file1_0, file1_1

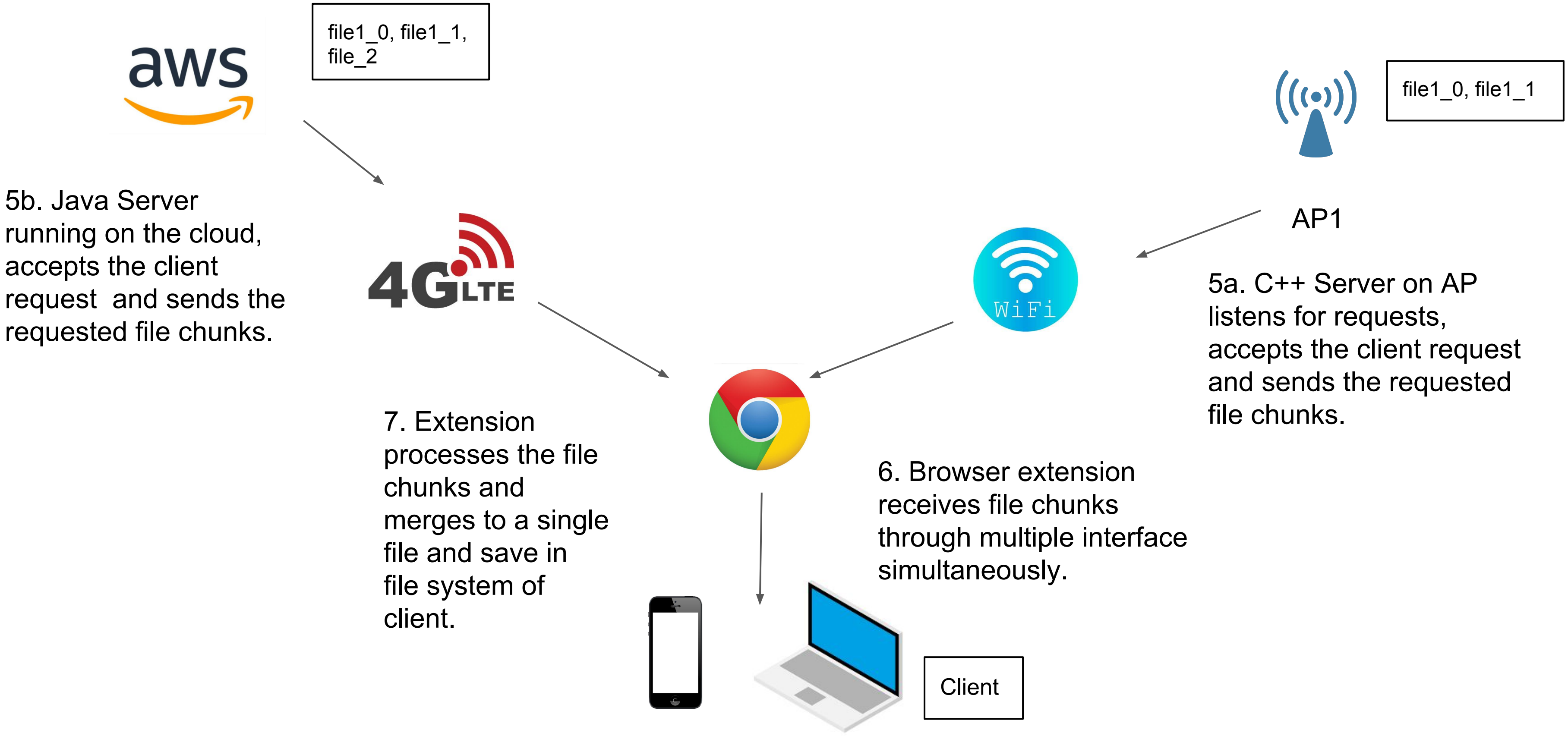
AP1

4b. Requests for file chunk file1_2 from the cloud server through other available network interface.
Eg: Cellular, Ethernet

4a. Requests for file chunks file1_0, file1_1 from the access point it is connected to through WiFi.

Client

Scheme (continued)



5b. Java Server running on the cloud, accepts the client request and sends the requested file chunks.

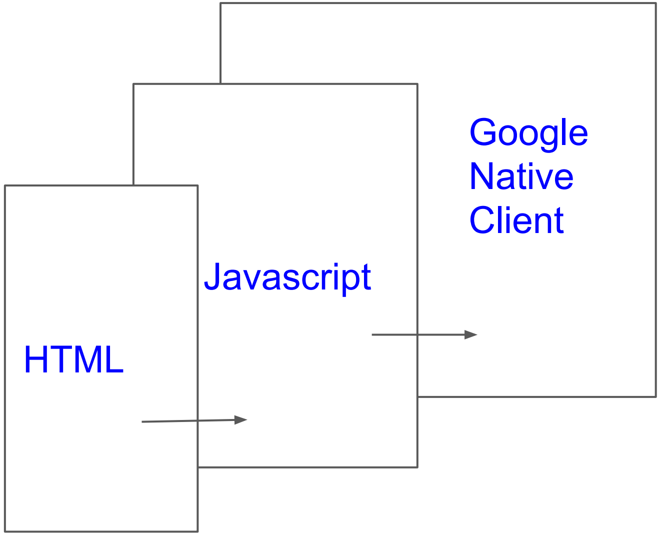
5a. C++ Server on AP listens for requests, accepts the client request and sends the requested file chunks.

6. Browser extension receives file chunks through multiple interface simultaneously.

7. Extension processes the file chunks and merges to a single file and save in file system of client.

Implementation

1. Web application



An HTML - Javascript web application is implemented along with a native client module.

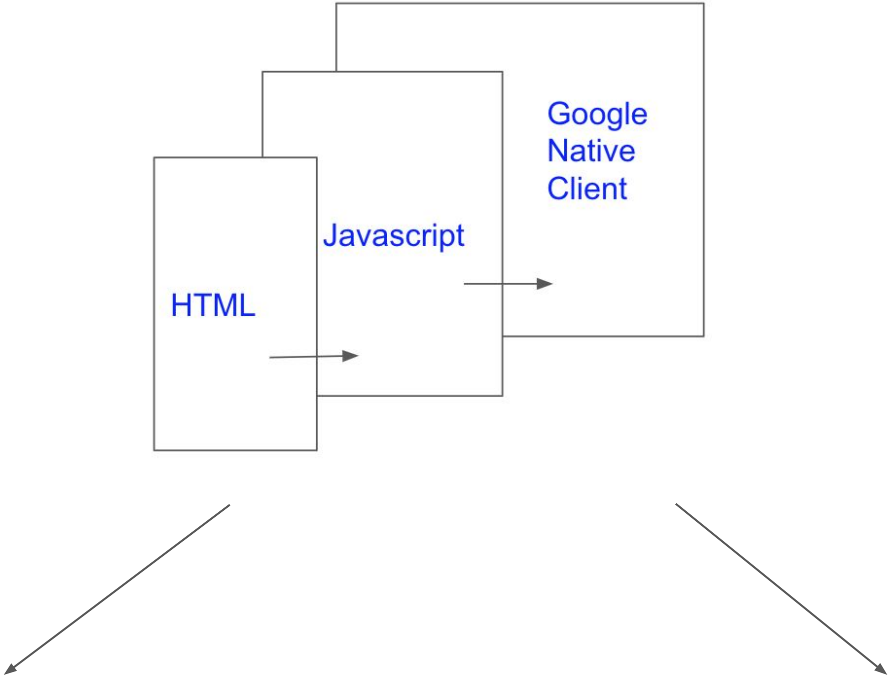
The Google Native Client is specific to Google Chrome.

Native Client is a sandbox for running compiled C and C++ code.

Why Native Client?

- Javascript do not have socket APIs which can be used to access a particular network interface for security reasons.
- Native Client that uses C/C++ has APIs to access the required network interface.
- Command POST_MESSAGE is used to call Native code from Javascript.
- Thus, Native client has the inherent security provided by Javascript.
- The APIs enable to choose a particular interface for transfer of particular data.
- It accesses multiple interfaces simultaneously by using multiple threads.

Implementation (continued)



Send the code everytime upon receiving a file request through an URL.

Bundle the code as an extension for the web browser.

Implementation (continued)

2. Server on Access Point

OpenWrt operating system is installed on all the access points.

C++ UDP Server is setup on the APs that listens for request.

Capable of serving files stored on cache of APs.

3. Server on Cloud

An Ubuntu operating system with Apache server is installed on a machine running on Amazon Web Services.

A Java UDP Server is setup on cloud that also serves files in addition to AP.

The files can also be downloaded individually from cloud server when no AP is available.

Experiment

- Successfully implemented server on one Access Point and on Cloud.
- The browser application sends file request to remote server via cellular connection (using USB tethering).
- After receiving the response, native client starts two threads for each interface
- It can successfully bind two sockets to two interfaces using the two threads.
- The second interface uses WiFi to connect to the local Access point..
- The client thus forms two parallel connections and sends chunk request through each.
- After processing the request, each server sends file chunks through the UDP socket.
- Native Client receives the incoming chunks simultaneously and stores on browser cache.

Conclusion

- The project explores a novel idea of multisource and multipath content transfer using multiple network interface.
- The idea is successfully implemented into a prototype browser application with the use of Native Client module.
- It has been a great learning experience for me by getting introduced to socket programming and working on a practical problem.

Future Work

- Analyse the performance of the system through various metrics such as download time, throughput.
- Compare the download time of the two network connections.
- Add streaming functionality and develop scheduling algorithms.

THANK YOU