IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 5, Issue 4, April 2016*

# Optimization of Bandwidth for IoT

**Roopa M[1], Nandini B M[2]**

PG Student, Dept. of Information Science and Engineering, The National Institute of Engineering, Mysore, India[1]

Assistant Professor, Dept. of Information Science & Engineering, The National Institute of Engineering, Mysore, India[2]

**Abstract**: Smart phones are most fascinating technology that everyone uses in today's life. Mobile phones are bundled with lot of features. Smart phone users are using internet in large volume. These devices using the wireless network consumes huge bandwidth when uploading or downloading data whereas performing the same operations using the desktop computer would be in the acceptable range. The service providers need to cater to the present growing need of bandwidth requirement without investing on new technology and also without negotiating on users expectations in order to survive in the market. Providing such huge bandwidth to each user is a terrible task. Hence, we propose bandwidth optimization algorithm using cache coherence which enables the service provider to cater to the growing needs of bandwidth requirement.

**Keywords**: Smart phones, Git, Version controlling systems, Mobile Networks.

## I. INTRODUCTION

Smart phones are becoming very popular. Besides making phone calls, they provide directions through GPS, take pictures, play music and keep track of meetings, appointments and contacts. Through the installation of Apps, the list of possible smart phone uses multiples by tens of thousands and grows longer every day. With this growth, internet is becoming ubiquitous on these devices.

The users are able to access internet in a go. As technology is advancing in terms of communication speeds, it's becoming difficult for the service providers to invest at the same pace. Hence there is a necessity of a methodology for the service providers to catch up the new trends with existing technology. Versioning of file allows us to keep the changes made to the file and recalling it at the later time. While developing a code, one might make several changes and want to retain each changes made to the code before making a new change, in that case Versioning Control System is the best choice.

It allows the user to revert back the changes to previous state, compare changes and also check what modifications were made and by whom. GIT[3] is distributed version control system. It is fast and small in size.

Each client contains the mirror of repository. Due to its distributed nature of version controlling and high compression levels, it is suitable for optimization. We use this concept for optimizing the bandwidth in mobile network.

Instead of copying the whole repository at the client, we use the technique used in EDGE[6] and GPRS[5] which copies only the changes made to the files. In this paper we propose a algorithm which is more efficient and simple that calculates the difference of two versions and merges only the delta obtained by comparing two versions to the file present into the cache and produce the resultant file to the user.

## II. RELATED WORK AND METHODOLOGY

This section describes the problem statement and proposed method to overcome the problem.

### *A.* Problem Statment

When the file request is made, cache is checked for the consistency and data availability. If the data is present then it is fetched from the cache else the requested file is downloaded and copied into the cache. This unnecessarily consumes lot of bandwidth. In addition to that, lot of traffic is also introduced.

### *B.* Proposed Solution

Downloading huge amount of data consumes lot of bandwidth. Hence we propose an algorithm which downloads the entire file and maintains a copy in the cache with its version when it's requested for the first time. Whenever the file is requested again, the cache is checked for the consistency and data availability. If the cache contains the latest version then the data is fetched from cache else the difference of two versions i.e; the delta is extracted, compressed and transmitted to the client which is decompressed and merged into the cache before producing it to the client and updates the version with the latest one at clients cache. This will reduce the bandwidth consumption effectively as the data being transmitted through the medium is very less.

### *C.* Proposed Design

Typical system architecture is shown in Fig 1. Ideally when the http request is made to the server, the server will process the request and send the response. This response may contain data which does not have much modification. Existing cache algorithm uses timestamp. Using this timestamp it checks whether there are any modifications or not. If there are no modifications, then contents from the cache is produced to the user. If there are modifications, then the modified file will be copied completely into the cache after downloading it from the server. This process consumes huge bandwidth since

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 5, Issue 4, April 2016*

whole file is being copied again even though modifications are very less.

Hence, we propose an algorithm in which upon detection of modifications, it calculates the difference between the versions present at the server and merges only the modifications.

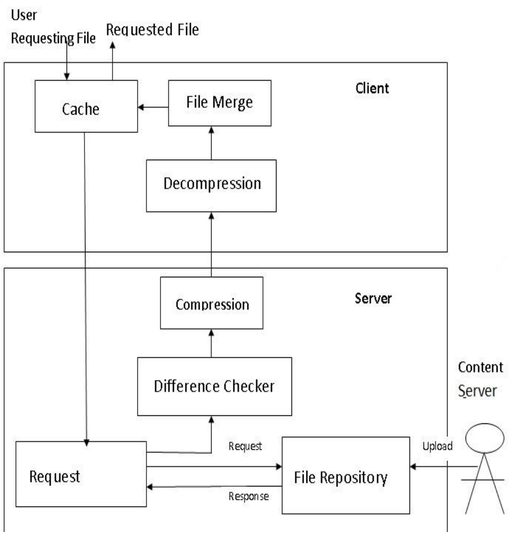

Fig. 1 System architecture for bandwidth optimization

The vendors and developers maintain their own repositories in which they maintain the versions of the file. These repositories of Git[2] and versioning concept[7] are used in the mobile communication for optimizing bandwidth. Client and the server both have to maintain the versions. If cache is up-to-date then same data can be used by the client. If there is a mismatch in the versions then, the difference is downloaded which is in compressed form. This downloaded data has to be decompressed and merged with the file present in the cache and produce to the client updating the version accordingly. This increases the performance by reducing bandwidth consumption and also compression at the server and decompression at the receiver avoids traffic.

The content server uploads the data and maintains the file repository to store the versions. Client also maintains the cache to store the fetched files along with the versions being fetched. When the client request for the file whose versions mismatch, the difference checker computes the differences between the two versions, compress the delta produced and transmits the compressed result to the client. At the client end the received data is decompressed and validated and changes are merged. The version gets updated in cache. This reduces the bandwidth consumption as the data being transferred over the wire is reduced effectively. In Fig2 each object interacts with other in sequential order through the messages. The bandwidth utilization of the existing versus the proposed system is shown in Fig3.

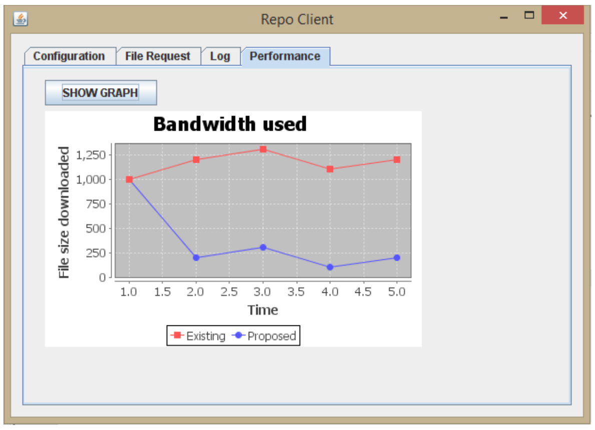

Fig. 3 Performance graph of Existing vs Proposed

*D.* Test Cases

TableI gives the test cases used to verify the system's functionality and the expected and actual results obtained.

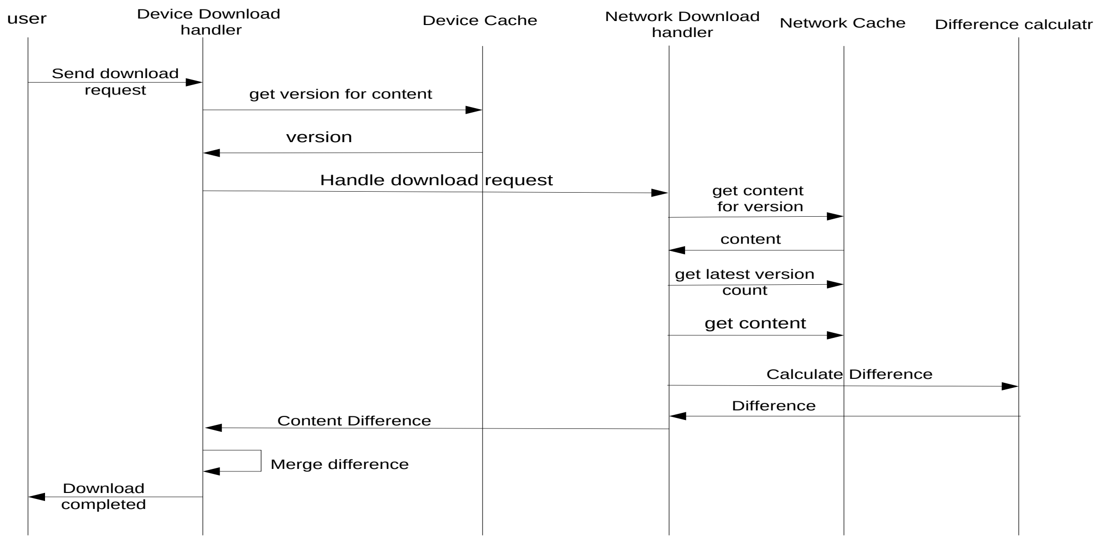

Fig. 2 Time Sequence Diagram of Operation

TableI: Test Cases

| S. No. | Test Case | Input given | Expected Result | Actual Result |
|---|---|---|---|---|
| 1 | Server accept file request | File Name | Accept file request | File request is accepted |
| 2 | Server receives request for new file | File Name | Entire file to be compressed and sent to the user | Entire file compressed and sent to the user |
| 3 | Server receives request for file whose versions mismatch | File Name | Only modifications to be sent in compressed form to User | Only the delta is sent in compressed form to the user |
| 4 | Server receives request for file whose version remain same. | File Name | File not to be sent. | File is not transferred. |
| 5 | User inputs invalid file name | File Name | File is not available at the server message has to be displayed. | File is not available message is displayed. |
| 6 | User requesting file for the first time | File Name | Entire file to be received by user. | Entire file is received by user. |
| 7 | User requesting file already present in its cache and are of same versions | File Name | File should not be received. | File is not received |
| 8 | User requesting for a file whose versions mismatch | Fie Name | Only modified data should be received in compressed form and the version has to be updated with the latest one in the cache | Only modified data is received in compressed form and the version is changed to the latest version.. |

## III. CONCLUSION

Today, mobile phones are being popular extensively. Earlier it was only used for calling purpose, but nowadays, most of the transactions over internet are made using mobiles only. Downloading and uploading huge data consumes lot of bandwidth. Supplying this huge bandwidth to the growing need of smart phone user is really a big challenge. Providing the required bandwidth without compromising the user acceptation and also not investing much on increasing their capacity is really a challenge! To deal with this challenge, this paper provides the solution for this problem by using a Bandwidth optimization algorithm which is based on cache coherency. The purpose of this paper is to provide more bandwidth with no expansion of capacity and also without compromising user's expectations and also reducing the traffic by compressing and decompressing the contents at server and user end respectively. The solution proposed can be used to any device that is hooked to the internet.

## REFERENCES

[1]. Performance Optimization of Big Data in Mobile Networks, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.
[2]. S. JC Hamano, "GIT – a stupid content tracker", Proceedings of the Linux Symposium 2006, Ottawa, Canada
[3]. Scott Chacon, "Pro Git", A press publication, Section 4.1 to 4.11 and Section 9.2 to 9.4
[4]. Wikipedia "Smart phones" https://en.wikipedia.org/ wiki/ Smartphone
[5]. Wikipedia "General Packet Radio Service" http://en. wikipedia.org/wiki /GPRS
[6]. Wikipedia "Enhanced Data Rates for GSM Evolution", http://en.wikipedia.org/wiki/EDGE
[7]. A. Ramaprasath, K. Hariharan, A. Srinivasan, "Cache Coherency Algorithm to Optimize Bandwidth in Mobile Networks", Springer Verlag, Lecture Notes in Electrical Engineering, Networks and Communications, Chapter 24, Volume 284, 2014, pp 297-305