A Framework for combining bandwidths of nodes in a network

Priyank Jain
Department of Computer Science and Engineering
Jaypee Institute of Information Technology
Noida, India - 201307
priyank4893@gmail.com

Dhananjay Tomar
Department of Computer Science and Engineering
Jaypee Institute of Information Technology
Noida, India - 201307
undercutspiky@yahoo.co.in

Abstract— Till now many efforts have been made to create a tool that can help in increasing the download speed of internet and help the users download any form of data from internet. But the fact that users over a network with limited bandwidth might be able to combine their bandwidths has never been considered while creating such tools. In this paper, we propose a framework which can help users to increase the download speed by helping each other as a group. This tool would be of great help in structures where computers are connected with each other via fairly high speed LAN and each computer has access to internet, like university hostel or some business organization.

Keywords— combine bandwidth; fast download; overcome limited bandwidth

I. INTRODUCTION

Over the past few years, we have seen large number of download tools like Internet download manager [1], Download accelerator plus and many others. All these tools do a great job in handling multiple download requests and come with top-notch multithreading which helps in making the downloads stable. But none of them have any option to integrate or to make use of more than one internet connections available.

In this paper, a framework has been proposed to combine the speed of multiple internet connections and utilize the maximum bandwidth available to a group of users. This framework when implemented, shall give good results.

The remainder of the paper is organized as follows. Section II describes the scenario where this tool will be of great use and also states why this kind of tool is needed. Section III consists of related work done by others whereas Section IV discusses the design scheme and the algorithm proposed by us.

II. UTILITY

In the structures like university hostels, companies, all the systems are connected through a local network (wired or wireless) and every system has connection to internet through the same network. In such situations even though the bandwidth of network is pretty high, but bandwidth available to each IP is usually limited. This sometime makes downloading large amount of data very slow and thus frustrating for a user. But a group of users can help each other increase the download speed by dividing the data to be downloaded and combining the bandwidths available to each one of them. Also in some university hostels every student has a limited amount of data that he can download in a single day, say 1 GB, so by dividing the downloading task with peers one can download a file bigger than 1 GB without any problems. In this paper we show how that can be achieved. The fig 1

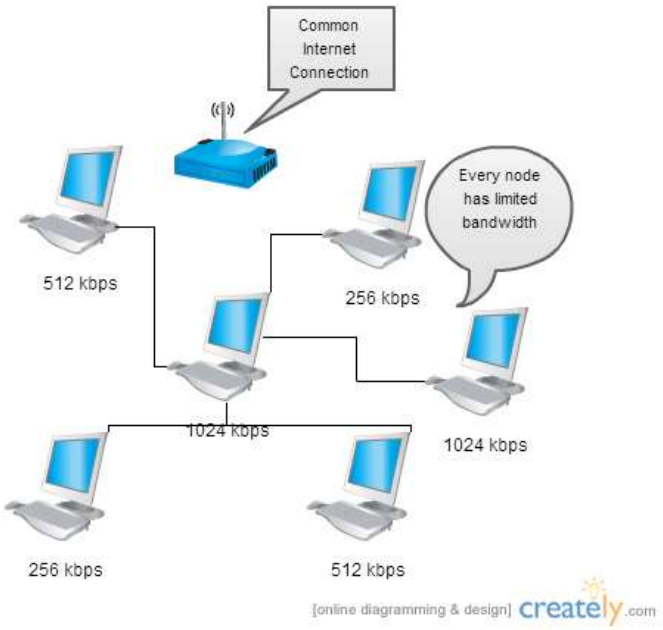

Figure 1 All the computers are connected to internet via same router

presents a scenario where this framework can prove to be of great help.

This framework will enable the connected systems to download data together by dividing the amount of data to be downloaded amongst the connected systems and later merging all the data on the main server machine. We call *server machine* as the system which has requested the download and we refer the rest of the nodes in the group as *client machines* which help the server in downloading the data. The client machine further increases the total download speed by parallelizing the task using threads thus utilizing its complete bandwidth.

III. RELATED WORK

Till now, not much work has been done in this area. Connectify dispatch [2] is one popular example of such product. It is a software load balancer that lets you use all the internet connections available at a time. It combines the bandwidth of Wi-Fi, 4G, 3G or ethernets and it appears as single bandwidth. But to use this, you need a separate Wi-Fi card for each additional Wi-Fi network you'd like to join as it works on a single system and does not make use of multiple systems.

How this paper's approach is different from Connectify dispatch:

In the scenario that has been presented above, there is a single internet connection available to multiple users as opposed to multiple connections available to a single user and we also have additional systems available on the common shared network to take advantage of. So, we developed a framework which can help to utilize the bandwidth of a network with the help of proper communication between the computers over that network. Detailed design and approach has been discussed in next section.

IV. DESIGN

The framework involves number of phases ranging from connection with all hosts on the network to downloading the data and merging it.

First step involves making a connection with all the hosts on the network that are online. When the software starts, it broadcasts a UDP packet on the broadcast IP address [3] and a specific port to check the live nodes (computers) on that LAN. The nodes having the same software running respond to the broadcast and send the bandwidth available to them. Thus the software gets the list of active IP addresses and also the amount of bandwidth (download speed) available to each host at that time.

Then the user enters the URL of the file that he wants to download. The software on user machine (also called server machine) sends request to the connected hosts (the list of active/live nodes) requesting them to help in downloading the data. The users respond with "yes" or "no" indicating whether they want to help the server machine in downloading the data or not. The software then checks whether the server hosting the to-be downloaded URL allows *byte-range requests* [4] or not by sending head request to the server. *Byte Range Requests* allow partial content requests thus allowing multiple streaming to a file.

If server does not allow ranged requests then server machine sends request to the host with maximum bandwidth to download the data. In case the server itself has maximum bandwidth, then it itself will download the file.

If the server allows byte-range requests it means several request could be made to download parts of the file.

Problem 1: The download speed might be different on all systems or some systems might already be consuming a part of their available bandwidth.

To solve this issue the software intelligently divides the amount of data to be downloaded by each machine who accepted request to help, according to the ratios of their current download speed. System with maximum speed is given maximum data to download and system with minimum speed is given least data.

Now since the host server allows more than one connection to a single file stream from any client IP address, software on each machine can start several threads to download multiple parts of file simultaneously.

Problem 2: If the data is huge, then after completion of downloading, transfer of data to server machine through LAN can be time-consuming.

Each thread on every node sends its data as soon as it downloads the amount assigned to that thread. This allows efficient use of network as opposed to a node sending the whole data after completely downloading it which would delay the process and keep server machine idle for long time. As soon as the server receives the data from any client, the software keeps on parallely merging the data by writing all the data at correct positions to a new file. Since the server knows the nodes, the exact part of file it was assigned to download and also how data was distributed among threads on each node, hence it knows which bunch of data needs to be written at which position in the file.

Problem 3: What if any internet connection on a node goes down or its download limit is reached during download?

As soon as any of the two situations are met, that particular client first tries to reconnect the hosting server. If connection is re-established successfully, it continues the download. But if the re-connection fails it sends a message to server machine containing information about amount of data remaining, starting and ending byte of that bunch of data. Now to server can forward the downloading request to any node who has completed its task and if no such node exists then to the node with maximum download speed.

We have depicted the whole process as a flowchart in fig 2. Below is the formula used for division of chunks of file to be downloaded among the nodes who accept to help in downloading the file.

Main system (the system that makes the request to download the file) is also a part of connect nodes here

Data ← total file size to be downloaded

Let *S* be the array of speeds of connected nodes

S_i ← speed of i^{th} connected node

N ← number of nodes who accepted download request

List of connected nodes is sorted in ascending order of their speeds

Let *D* be array of Data to be downloaded by each node

Then

i ← 1

sum ← $\sum_{i=0}^{N+1} Si$

for every data $d \in D$ {

if i < N+1:

$$d \leftarrow \frac{Data}{(\frac{sum}{Si})}$$

else:

$$d \leftarrow \frac{Data}{(\frac{sum}{Si})} + (Data)\ mod\ (\frac{sum}{Si})$$

i ← i + 1

}

At every node, there are multiple threads to download data.

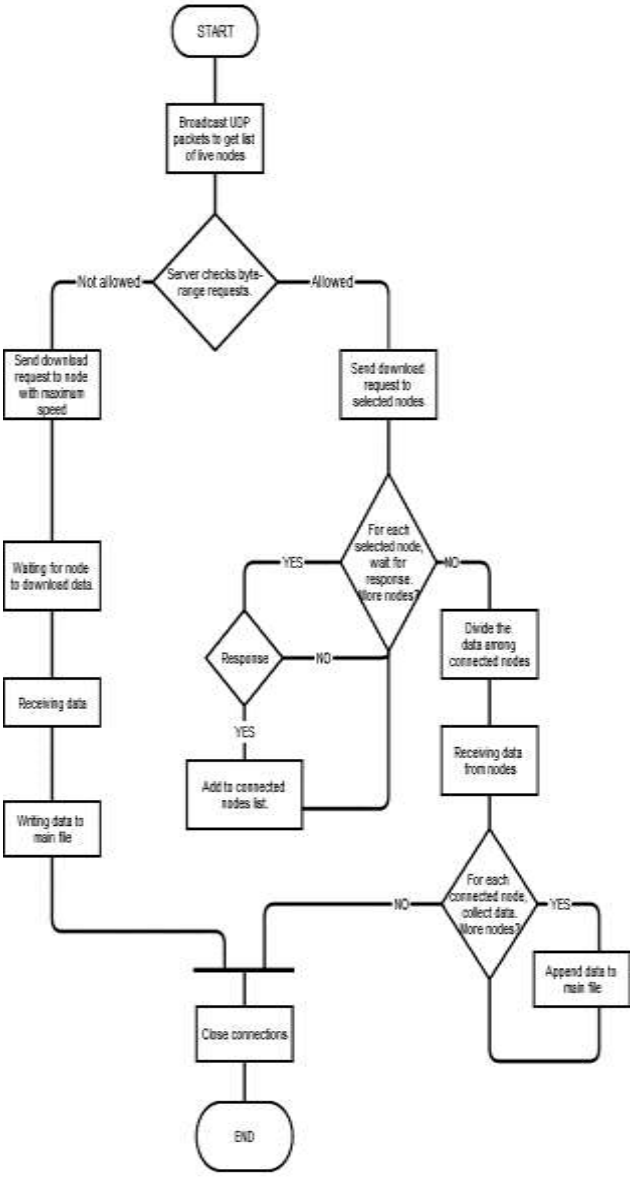

Figure 2 Process Flowchart

Below is the formula to divide it among the threads equally.

Data ← total file size to be downloaded by that node
N_t ← number of threads
Let *D* be array of Data to be downloaded by each thread
Then
i ← 1
for every data $d \in D$ {
if i < N_t:

$$d \leftarrow \frac{Data}{(\frac{sum}{Si})}$$

else:

$$d \leftarrow \frac{Data}{(\frac{sum}{Si})} + (Data)\ mod\ (\frac{sum}{Si})$$

i ← i + 1
}

V. FUTURE SCOPE

The framework presented can allow efficient use of bandwidth available to an organization. The framework can further be integrated with utilization of other internet sources that user might have available like 3G on cell phone or a data card. Also with the help of an appropriate dynamic segmentation algorithm it might be possible to add a new in between the download process and give it some amount of data from node with least bandwidth or node which has largest chunk data remaining to be downloaded. Such an algorithm would also allow any node that has due to some reasons has now low bandwidth available, to divide its data among the nodes that have completed their download.

VI. CONCLUSION

A new framework for fast download of data in a network with nodes having limited bandwidth has been proposed and through our discussion we have shown how such a framework is possible and how it can help increase the bandwidth of any user desired. Once implemented, this could prove to be of great help.

VII. REFERENCES

[1] Internet Dowwnload Manager, http://www.internetdownloadmanager.com/
[2] Connectify Dispatch, http://www.connectify.me/dispatch/
[3] RFC 919, Broadcasting Internet Datagrams, J. Mogul (October 1984), http://www.ietf.org/rfc/rfc919.txt.pdf
[4] RFC 2616 – Section 14, http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html