CDN Project

Introduction:

In this project, we overcome the latency issue in downloading web content. The problem is solved using content delivery network in which servers are placed in different regions around the globe and user is served from nearest server. While having content servers, it is challenging to maintain efficient connectivity between servers and upload limited amount of data to save cost. This challenge is solved using distance vector routing and caching data efficiently such that least amount of storage is used on servers. We have defined two different caching mechanisms here.

Design:

The project consists of creating identical CDN server and Content server in all regions. There is also a proxy server that forwards requests from user browser to one of the randomly fixed CDN server. The proxy and content server are simple HTTP servers.

Since the CDN server must control routing and content handling, it consists of three processes for reachability, routing and http server. All three processes are running parallel using Threading technique. Threading is also used to handle multiple requests simultaneously.

The reachability process is used to discover neighboring nodes and calculating delay in reaching them. This calculated delay is used by routing process to find the optimal path to specific node. The route with minimum delay is selected as best path. This information is collected by sharing neighboring routes detail with each other.

Whenever a HTTP request is received, the http server process will first try to serve the content from its internal cache. If its not found there then it is forwarded to next-hop in the route to destination node. The content is cached based on the defined mechanism.

Implementation:

The project is implemented using Python version 3. Some libraries are imported in the code as mentioned in README file to perform tasks such as HTTP request handling, generation HTTP request, URL parse and threading.

The code format in each of the different servers (proxy, content & CDN) consist of following hierarchy:

1. Importing modules
2. Defining constants
3. Creating methods
4. Running methods through main function

The proxy and content server are implemented using Python HTTPbasehandler library to serve HTTP requests. Also, threading is used to support multiple requests. The only difference is that proxy server forwards the request to the mentioned CDN server while content server replies with content. Both servers are configured to handle HTTP GET requests only.

The main part of the project lies in CDN server code. This server handles three process simultaneously as described above but their details is as follows:

1. Reachibility Process (pingpong() method):

The CDN server sends HTTP request with 'ping' as path to directly connected nodes mentioned in conf file. Upon the reception, it notes their location name (geo-tag) and calculates delay to them. This process is done concurrently with 30seconds interval if all nodes replied or else with 5second interval to rigorously monitor dead nodes.
New-feature: If previously live node (replying node) is dead (not replying) then the link is assumed to be disconnected and is given maximum link delay to stop forwarding in that way.

2. Routing Process(dvr() method):

The routing process starts with updating routing table with latest link delays of neighboring nodes. The routing table is converted as data payload and is send to all the live neighboring nodes. All the routing messages are exchanged using HTTP POST method. New routes are added or updated when routing message from neighboring node is received. Routes are added/updated using following logic:

```
Link_delay to be added = link_delay received + link_delay to that neighboring node
If [destination_node] == [local_node]:
        Omit the route
Else If [destination_node] not in ROUTING_TABLE:
        Add the route
Else If [link_delay to destination_node] < link_delay in ROUTING_TABLE:
        Update the route
```

This process runs in loop with 30seconds interval. Also, the process is not started with other processes but after 5seconds interval to let the nodes populate neighbor table.

3. HTTP Server Process (HTTPRequestHandler() method):

The server processes GET and POST methods only. The POST method is only used to serve routing process messages with 'dvr' as path.
The GET requests are served with following logic:

```
If URL_path in local cache:
        Serve from cache
Else If URL_HOSTNAME == local GEO_TAG:
        Send request to local content server
        Cache the content
Else If URL_HOSTNAME != local GEO_TAG:
        If URL_HOSTNAME in ROUTE_TABLE['GEO_TAG']:
                Forward request to next-hop
                Send after delay
                Cache the content based on the scheme
Else If URL_PATH == '/ping':
        Send reply with 'pong' as message
        Include localnode GEO_TAG and address in header
```

All the incoming HTTP request and responses are logged in file and printed on terminal also.

The content is cached on the CDN server based on the scheme 1 or 2.

For scheme 1, content is cached every time.

For scheme 2, content is cached only single time and a header field ‘Cached’ is added when the content is cached. The content is not cached if the response contains header field named as ‘Cached’.

Evaluation:

The whole project is executed on single instance of Ubuntu 16.04 LTS OS and on 6 instances on Amazon EC2 platform. The project successfully executed on both the scenarios. Its strange that there is more latency when running all the CDN servers on single OS. I guess this may be due to the limited resources on that machine.

Conclusions:

The idea of this project is to make aware of how Content Delivery Networks (CDN) works and what are the issues in distributed networks where something wrong in one node affects the output on other. Also, we observe the difference of latency when content is cached and not cached.