Implementation of ALTO assisted HTTP protocol in dynamic P2P systems

By Group 3

What is ALTO?

How is ALTO used in P2P networks?

ALTO (Application Layer Traffic Optimization) allows clients connected in a network to obtain information about paths to other endpoints in the network.

Typically, it is used to identify the lowest-cost location for a client to send/receive data to/from.

ALTO servers collect information of a network's topology, and uses that information to provide a route between the clients. The client first queries the ALTO server for the information and then connects with the end point with the lowest cost.

ALTO can be used with P2P networks, for example BitTorrent. Implementing ALTO would lead to better performance of such P2P networks and also would reduce utilization of the network infrastructure.

System Design

We intend to use Mininet for our project.

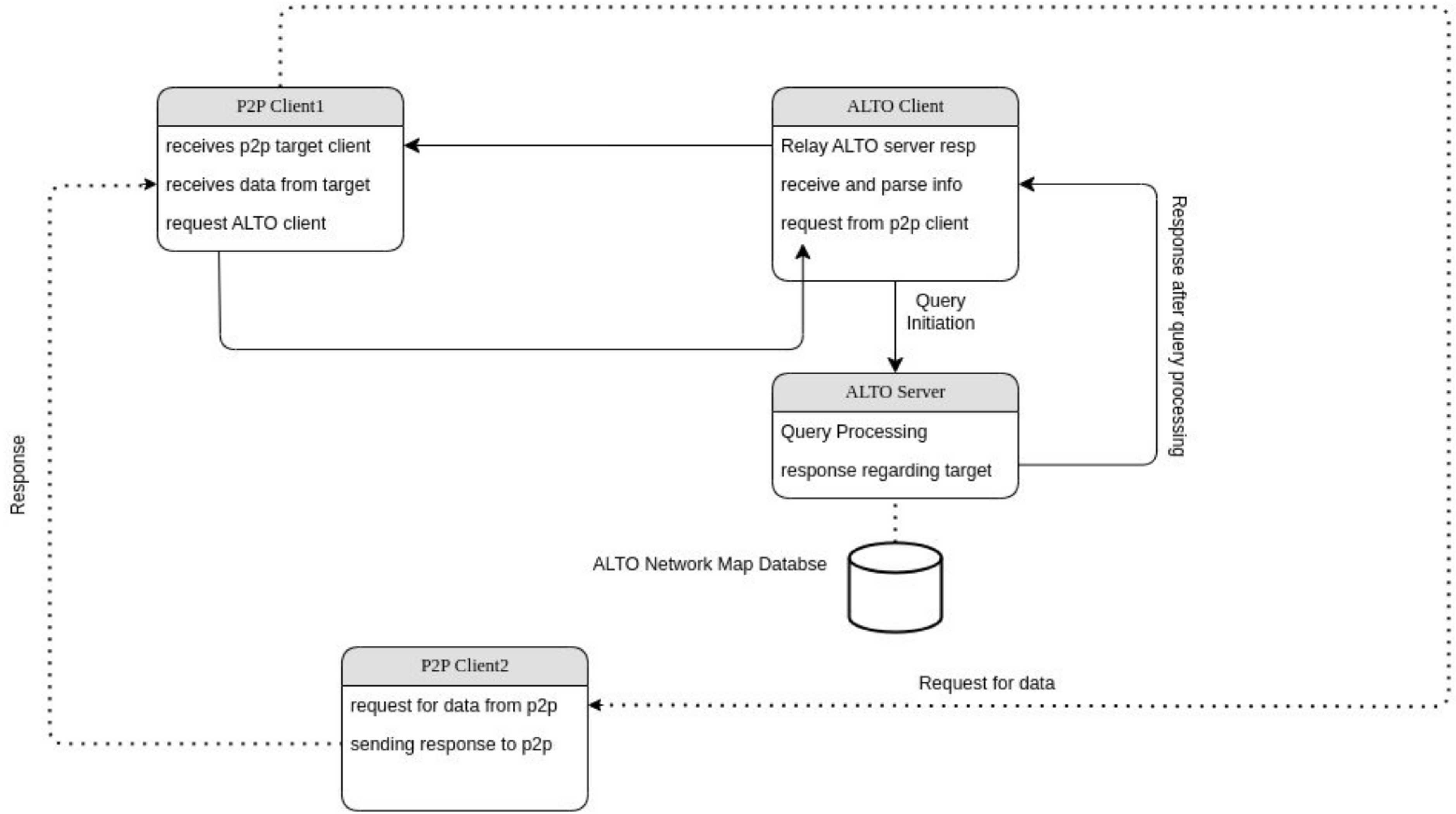
We intend to implement ALTO over HTTP protocol.

The environment will emulate a dynamic P2P network consisting of nodes having different behaviours such as:

- ALTO servers
- P2P clients
- Switches/Routers

ALTO servers will contain a network map database, which stores information about the underlying network such as the topology, resource availability, and other metrics. Using this database, it will assist P2P clients in making optimal decisions

P2P clients are just nodes that would help us emulate data transfers in a P2P network.


P2P Client1
receives p2p target client
receives data from target
request ALTO client
ALTO Client
Relay ALTO server resp
receive and parse info
request from p2p client
Query Initiation
Response after query processing
ALTO Server
Query Processing
response regarding target
ALTO Network Map Databse
Response
P2P Client2
request for data from p2p
sending response to p2p
Request for data

Topology Implementation

- The information regarding hosts(name,ip), switches(name,dpid) and links(source,destination,bandwidth,delay) are stored in a json file named topo.json
- Similarly there is topological information about individual elements such as peers in peers.json and ALTO server in server.json
- Currently we have only one ALTO server in the entire environment so we do not have any ALTO server discovery requests.
- We setup the mininet by executing the mininet.py file, in which the topology information is loaded and a virtual topology is created.

Peers Implementation

- The peer creates a separate thread to listen to user commands/requests
- The peer has to "register" with the ALTO server, "unregister" when it goes offline and sends a HTTP request to the server to "GET" the best peer to connect to for downloading a file.
- In-order to "register" on the ALTO server, the peer sends a an HTTP POST request to the ALTO client and similarly it sends a HTTP POST request to "deregister" from the server
- The peer also has a "get_file" request to download a file from the network
- A "receive file" request is also available in which the user listens on socket "1". Opens a File and write in blocks of 1024 bytes. Connection is closed if file is received successfully
- Main Problem for peers is finding best peer

ALTO Server Implementation

- The ALTO cost-map is implemented using the **networkx** library to create a cost-map graph
- The static topology map for the cost-map graph is present in topology.json
- The ALTO server accepts HTTP requests to “register”, “unregister” and “get_best_peer” to connect to (basically the peer with least cost)
- The ALTO client which sends requests on behalf of the peers is built into the peers
- All communication between server and peers happens through HTTP
- The ALTO server uses min_hop_count, bandwidth and delay as metrics to calculate the cost between two links (the cost function which incorporates all these metrics is not done)

Problems Faced

- The logic for file-management among peers
- The logic for incorporating the metrics into the cost equation for calculating costs of different links
- The logic for dynamically changing the costs in the cost-map implemented as a graph using **networkx** library

THANK YOU!

BY GROUP 3