# Homework 1: Reverse proxy

## Due: 3/15/2020

In this homework, we will write an application-level reverse proxy system. Conceptually, this setup has many elements of a packet switching, reverse proxy and load balancing system.

In a typical reverse proxy system, there are two types of entities. First, we have the reverse proxy that keeps track of different devices that are connected and accessible through it. Second, the devices or nodes that connect to the reverse proxy. In our setup, there are two types of nodes -- client, which needs to send messages to a server, which is the second type of node. The reverse proxy and client/server are implemented in Python using basic socket interface.

In our setup, we have a single reverse proxy and multiple clients and servers. The clients send text messages to any server identified by a specific privacy policy and the responding server sends to the client a SHA1 hash of the message. The client can then check if the server received the message correctly and display an appropriate transmission status message on the terminal.

## The Packet (Message)

Network packets are typically sent in binary format, which are compact. In this homework, we will send our messages in JSON format for convenience.

```
{
  "type": 0,          // 0 is a message from a client to a server
  "srcid": 999,       // source (client) id
  "privPoliId": 999,      // destination server's privacy policy
  "payloadsize": 999, // payload size
  "payload": "xyz"    // payload
}
```

## The Client

The client wants to send a message to a server serving a specific privacy policy. It knows the privacy policy of one many servers running the requested policy but does not know the ports on which these servers are running so is not able to open a socket to one of these servers. It uses TCP socket to connect to the reverse proxy and sends the packet to the reverse proxy expecting it to forward the message to the server.

Here is how to run the client:

```
  python client.py -id ID -revproc PORT -pkt PKTFILE
```

Replace ID with the client for the client. Replace PORT with the port number on which the reverse proxy is accepting connections. Replace PKTFILE with the name of the file in which you have the message in JSON format.

## The reverse proxy

The reverse proxy listens for incoming connections from the clients and servers on a well-known port.

When a server connects to the reverse proxy, the incoming connection is immediately followed by this setup message:

```
{
  "type": 1          // 1 is a connection setup message from a server
  "id": 999,         // id of the server
  "privPolyId": 999, // privacy policy of the server
  "listenport": 999  // port on which the server is listening
}
```

When the reverse proxy receives this setup message, it creates a message switching table that records the id of the server, its privacy policy and the port number on which the server is listening.

When the reverse proxy receives a message of type 0, it looks up in its message switching table for the servers serving the requested privacy policy, selects one of these servers, connects to the correct port number, and sends the message to that port. That is how the reverse proxy forwards the message from the client to the server.

The reverse proxy also has a load balancing functionality. If we have multiple servers serving the same privacy policy, the reverse proxy distributes the incoming client messages between those servers using round robin.

Example:

We have 5 messages requesting privacy policy **PP_1** and two servers (1 and 2) serving privacy policy **PP_1**. The messages should be distributed as follows:

- Message 1 goes to server 1
- Message 2 goes to server 2
- Message 3 goes to server 1
- Message 4 goes to server 2
- Message 5 goes to server 1

Here is how we run the reverse proxy:

```
  python revproc.py -port PORT
```

Replace PORT with a port number. This is the well-known port the clients and the servers will use to connect to the reverse proxy.

## The Server

The server listens for a message of type 0, which is a message from the client. It grabs the text in the payload and computes SHA1 hash of the message and sends it back to the client in the payload of the acknowledgment message. Here is what the message looks like:

```
{
  "type": 2,          // 2 is an ACK from a server to a client
  "srcid": 999,       // source (server) id
  "destid": 999,      // destination (client) id
  "payloadsize": 999, // payload size
  "payload": "xyz"    // payload
}
```

Here is how we run the server:

```
python server.py -id ID -pp PPID -listen LPORT -revproc RPPORT
```

Replace ID with the id number for the server. Replace PPID with the privacy policy id number for the server. Replace LPORT with the port on which the server listens for messages. Replace RPPORT with the well-known port on which the reverse proxy is running.

## Running the system

Setup the network by running the reverse proxy first, then server(s), then client(s). To make it easy to see how each system works, you can run them on different terminals.

The messages on the reverse proxy terminal may look something like this.

```
% python revproc.py -port 123


Running the reverse proxy on port 123


Receiving setup message from server id 100 privacy policy PP_1 port 124


Received a data message from client 50 privacy policy PP_1 payload: {.....}


Forwarding a data message to server id 100 payload: {.....}


Received a data message from server id 100 payload: {.....}


Forwarding a data message to client id 50 payload: {....}
```

The messages on the server terminal may look something like this.

```
% python server.py -id 100 -pp PP_1 -listen 124 -revproc 123


Server running with id 100


Server serving privacy policy PP_1


Listening on port 124


Connecting to the reverse proxy on port 123


Received a message from client 50 payload: {......}


Sending a response to the client 50 payload: {......}
```

The messages on the client terminal may look something like this.

```
% python client.py -id 50 -revproc 123 -pkt mypkt.txt


Sending message {......} to privacy policy PP_1 through reverse proxy running on port
123


Receiving a response from the server 100 payload: {.....}
```

## Submission

I