# GENI Mini Project

You are allowed to work in teams of 4. You are asked to investigate more deeply a certain aspect of networking experimentally, over the GENI testbed. The topic is up to you, though we list a few suggestions below. You are required to submit a brief 1-2 pages proposal that describes the question(s) that you plan to study experimentally and what your methodology will be. **The proposal is due on Gradescope on Thursday, November 19, 2020.** The instructors will provide feedback on your proposal, possibly suggesting decreasing or increasing the scope of the work appropriately. The final report should be published on GitHub and describe your problem, objectives, experimental methodology, results and analysis, along with a complete set of instructions that allows for the reproducibility of your setup and results. **The final report is due on Wednesday, December 9, 2020 (no late submissions will be allowed).**

***Basic Skills:*** *Students need to have basic knowledge of GitHub and should be familiar with command-line usage for all the projects.* **All final projects should be available on GitHub with the code and readme with clear deployment and usage instructions.**

## *Project Ideas:*

### ***Web Caching***

Design an experiment on GENI nodes where one can show the benefit of having a web cache, automate the process of retrieving objects through the cache and without the cache. Also, write scripts to generate results, e.g. latency with a cache and without a cache. Also, study different aspects of web caching, e.g. max-age, expires headers and recent research work on caching (see xCache, RFC).
**Skills:** Should know how to install and enable a web cache (ATS), how to direct a request to a cache (wget/curl), and plotting tools.

### ***Password Cracker***

Design a distributed system (similar to Hadoop), where a user submits the md5 hash of a 5-character password (a-z, A-Z) to the system using a web interface. The web interface with the help of worker nodes cracks the password by a brute force approach. Students need to implement the web-interface as well as the management service that will dedicate jobs to worker nodes. The web-interface and the management service can be on the same machine, but worker nodes need to be different machines. The systems should be scalable, i.e. you can add/remove workers on the fly. The management service should use the REST API or socket programming to communicate with the worker nodes.
**Skills:** CGI-bin, socket programming.

### ***Password Cracker (Serverless)***

Implement the password cracker above but instead of using dedicated worker machines, use serverless functions to do the computation.
**Skills:** CGI-bin, REST API, serverless platforms, e.g. OpenWhisk, containers.

### ***Image Recognition Application***

Implement an image-recognition service which has a web interface, where a user should be able to submit an image query and the service should use any of image recognition techniques, e.g. SqueezeNet, GoogleNet or any deep learning neural network which classifies the image and returns the answer to the user. You don't need to worry about training the neural network and can use pre-trained weights. The web interface and the recognition systems should be on separate nodes and you should connect them using socket programming or rest API.

**Skills:** CGI-bin, socket programming, usage of libraries/packages/models like SqueezeNet, etc.

## *Image Recognition Application (Serverless)*

This application would be similar to the above but instead of having to deploy the neural network on a bare node, you will deploy this as a serverless function. On receiving the image, the web server will invoke the serverless function for classification.
**Skills:** CGI-bin, REST API, serverless platforms, e.g. OpenWhisk, usage of libraries/package/models like SqueezeNet, containers.

## *Performance Comparison of various TCP versions*

You would be required to study different aspects of TCP flavors (e.g. Reno, Cubic, Vegas) in terms of delays, throughput, fairness, etc. You can also read up on new protocols and deploy them to compare with TCP for extra credit, e.g. Verus, Sprout.
**Skills**: command-line usage, experiment design, TCP flavors, deploying open-source code.

## *Multi-Path TCP*

Multipath TCP, as well as multipath QUIC, is an extension of TCP (QUIC) that allow using multiple links for the same connection. This improves reliability but can also utilize the throughput of all available links for the same connection. The first problem is especially relevant for smartphones that need to seamlessly transition between WiFi and the cellular connection. The goal is to study the proposed protocol and briefly summarize it. You will also need to find and run an open-source implementation and compare the multipath approach to the existing standard with respect to reliability or speed or both. A good starting point is to consult the wikipedia page: https://en.wikipedia.org/wiki/Multipath_TCP.
**Skills**: command-line usage, experiment design, TCP flavors, deploying open-source code.

## *μTP*

One problem that BitTorrent users experience is throttling of non-BitTorrent connections. A BitTorrent client can simultaneously talk to many peers. In the past, an individual TCP connection was used to talk to each peer. This led to other TCP connections (e.g., web browser) being limited in speed since typically, the bandwidth of a client is split equally between all its TCP connections. To overcome this problem, Micro Transport Protocol was developed. Your goal is to study and experimentally evaluate this protocol. A good starting point is to consult https://en.wikipedia.org/wiki/Micro_Transport_Protocol. The best way to run experiments would probably be to utilize this library: https://github.com/bittorrent/libutp.
**Skills**: command-line usage, experiment design, TCP flavors, deploying open-source code.

## *Other Ideas*

There are many GENI tutorials available, some of which we did. You can propose to extend any of them to study certain aspects in more detail. For example, this tutorial studies OSPF and you may want to study its scalability or compare it to some other routing protocol. Another tutorial studies firewalling and you may want to examine issues related to intrusion detection and prevention.