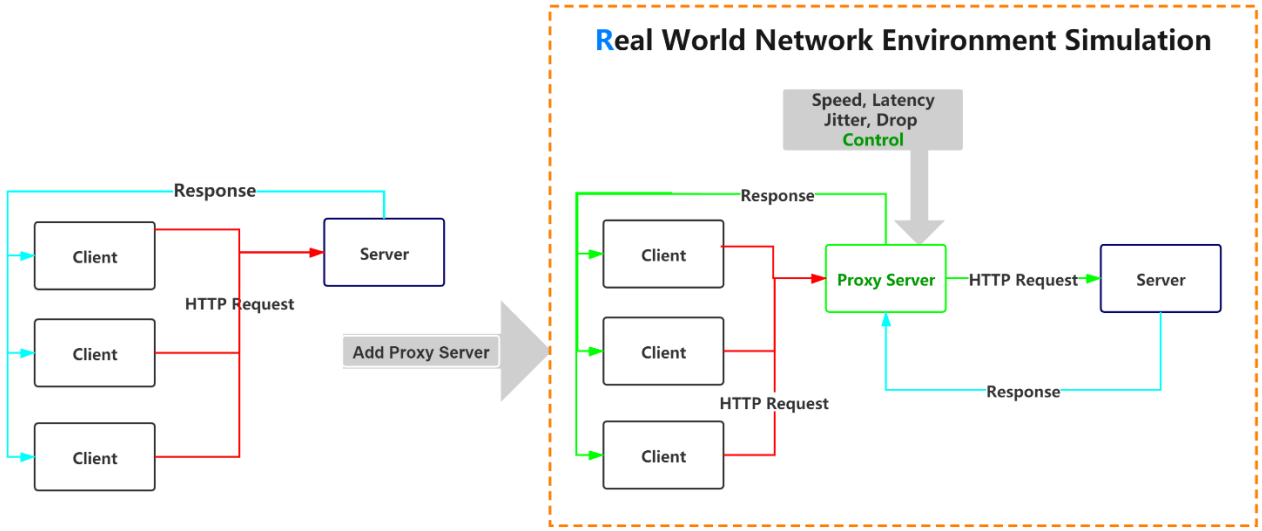
Introduction:

一个典型的 Web 应用与用户间的交互可以分为四个部分：一、用户（User）通过浏览器发送一个 HTTP 请求；二、服务器（Server）收到请求，生成一个 HTML 文档；三、服务器（Server）把 HTML 文档作为 HTTP 响应的 Body 发送给浏览器；四、浏览器收到 HTTP 响应，从 HTTP Body 取出 HTML 文档并显示给用户。

所以，简单的 Web 应用就是先把 HTML 用文件保存好，用一个 HTTP 服务器接收用户请求，从本地读取文件，生成 HTML 文档并返回给客户端。这就是 Apache、Nginx、Lighttpd 等一些常见的静态服务器的实现原理，本项目的服务器属于此类静态服务器。在现实中，静态服务器传输文档时常会遇到网络拥塞，服务器传输速率将会受到限制。并且，现实不稳定的网络环境将时不时产生网络抖动 Jitter，导致传输速率跳变，甚至出现断网。为了证实我们提出的新型 ABR 算法可以适应现有的网络环境，我们需要模拟多种不同的网络状况，以确定新型 ABR 算法能针对不同的环境进行较好自适应调节，使得 QOE 指标维持在一个相对较高的水平。

总体思路：

在 Server 端进行模拟需要改变服务器内置代码才能达到目的，需要对服务器的框架和主体代码做出较大改动。而我们通过增设一个代理服务器，使得原先客户端向服务器直接的请求变成向代理服务器发送请求，通过代理服务器再向原服务器发送原请求。代理服务器（Proxy Server）内置响应延迟时长（Response Latency）、传输速率上限（Transmission Speed）、网络抖动概率（Jitter）、断网概率（Disconnetion）等参数，以模拟各种典型的网络状况。代理服务器接受到原服务器的 HTTP 相应之后，再向客户端发送响应报文。



设计细节：

1. 使用 Python Tornado 作为 Web 框架,相比其它 Web 框架开发的 Server（Django, Flask .etc），Chaos-Proxy 并发性更强。
2. 使用异步框架搭建 Web Server。
3. 双模式（Simple mode && Advanced Mode）的代理服务器。

测试：使用 simple mode 对网络的基本环境进行了模拟，设置延迟时长（Response Latency）、传输速率上限（Transmission Speed）为(300ms，5000bit/s), (500ms，3500bit/s), (1000ms，2000bit/s), (2000ms，800bit/s) ，在服务器对应模拟网络状态下测试多个客户端并发请求不同字节客户端平均接收速率 V_{Client}。

此后，进一步组合上述几种参数的不同取值以模拟复杂的网络状况（如传输速率跳变、响应延迟时长跳变、随机网络抖动等）。

测试结果见 Pressure_Test_Results.docx