**WebPage Download Program**

1. Program takes object tree as input. Object tree shows dependency of various objects to be downloaded. It also has variables max\_connections, max\_tcp\_per\_domain and max\_objects\_per\_connections.
2. Initially we group all the objects to be downloaded as per the domain name. Then for each domain name we have a queue which contains the objects to be downloaded. We enqueue the objects which have no dependency issue.
3. Depending on max\_objects\_per\_connections, we launch GET request of the objects in the queue for each domain.
4. We observed that for most of the http\_response objects, server didn’t not send us content length. So we used transfer-encoding: chunked size to calculate how big the data is for each objects. As per the protocol, server sends us zero size chunk denoting completion of data.

b) Library Used:

Socket library in node.js

c) Page Load Time:

1. Our page load time is better compared to the browser. We have implemented pipelining in our programs as can be noted by max\_objects\_per\_connections. If this parameter is 1 it implies no pipelining.
2. Time with pipelining - 12 s (nytimes)
3. Time without pipelining - 8 s (nytimes)
4. This is better compared to our browser which was 50 s.

d) Pipelining with maximum 2 parallel objects per connection, along with 5 max connections per domain, with round robin scheduling across connections gives best page load time (7-8 secs for nytimes). More than 2 parallel objects result in extraneous/unexpected outputs from some servers, which corrupts object files.

e) There are two important pieces of information that can help us:

1. Provind object sizes in the parent object itself so that we can queue objects efficiently among different connections (i.e.: for two queues, [1mb,1mb], [1kb, 1kb] is worse than [1mb, 1kb], [1mb, 1kb]. but we cannot optimize this in present case.  
   Not using chunked encoding can also help, as we can atleast know the size once we receive header (a lot of servers didnt send a content length and relied on chunked encoding)